NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

GONÇALO ANDRÉ SANTOS ANTUNES

Bachelor in Computer Science

DEEP TEST TO TRANSFORMERS ARCHITECTURE IN NAMED ENTITY RECOGNITION

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon

⟨September⟩, ⟨2022⟩

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

DEEP TEST TO TRANSFORMERS ARCHITECTURE IN NAMED ENTITY RECOGNITION

GONÇALO ANDRÉ SANTOS ANTUNES

Bachelor in Computer Science

Adviser: Joaquim Francisco Ferreira da Silva
Associate Professor, NOVA University Lisbon

Examination Committee

Chair: Maria Armanda Simenta Rodrigues Grueau
Associate Professor, FCT-NOVA

Rapporteur: Francisco José Moreira Couto
Auxiliar Professor, FCT-NOVA

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
⟨September⟩, ⟨2022⟩

Deep test to transformers architecture in Named Entity Recognition

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.5) [26].

To family and friends.

Acknowledgements

Firstly, I want to express my sincere gratitude to Joaquim Silva, without whom the work presented in this thesis would not have been possible.

I want to thank all my friends for motivating me and being with me in this journey.

Finally, I would like thank my mother, father and brother for always encouraging me to do better, and be better. They taught me that you only need a seed and water to grow a tree .

"When you change your thoughts, remember to also change your world." (Norman Vincent Peale)

Abstract

Named Entity Recognition is a task of Natural Language Processing, which aims to extract and classify named entities such as "Queen of England". Depending on the objective of the extraction, the entities can be classified with different labels. These labels usually are Person, Organization, and Location but can be extended and include sub-entities like cars, countries, etc., or very different such as when the scope of the classification is biological, and the entities are Genes or Virus. These entities are extracted from raw text, which may be a well-structured scientific document or an internet post, and written in any language. These constraints create a considerable challenge to create an independent domain model. So, most of the authors have focused on English documents, which is the most explored language and contain more labeled data, which requires a significant amount of human resources. More recently, approaches are focused on Transformers architecture models, which may take up to days to train and consume millions of labeled entities.

My approach is a statistical one, which means it will be language-independent while still requiring much computation power. This model will combine multiple techniques such as Bag of Words, Steeming, and Word2Vec to compute his features. Then, it will be compared with two transformer-based models, that although they have similar architecture, they have respectful differences. The three models will be tested in multiple datasets, each with its challenges, to conduct deep research on each model's strengths and weaknesses.

After a tough evaluation process the three models achieved performances of over 90% in datasets with high number of samples. The biggest challenge were the datasets with lower data, where the Pipeline achieved better performances than the transformer-based models.

Keywords: Natural Language Processing, Named Entity Recognition, Machine Learning, Statistical, Independent Domain

Resumo

Named Entity Recognition é uma tarefa no Processamento de Língua Natural, que tem como objectivo extrair e classificar entidades como "Rainha da Inglaterra". Dependendo do objectivo da extração, as entidades podem ser classificadas em diferentes categorias. As categorias mais comuns são: Pessoa, Organização e Local, mas podem ser estendidas e incluir sub-entidades como carros, países, entre outros. Existem ainda categorias muito diferentes, por exemplo, quando o texto é do domínio da Biologia e as categorias são Genes ou Vírus. Essas entidades são extraídas de diferentes tipos de texto como documentos científicos estruturados corretamente ou um *post* da internet, podendo ser escritos em qualquer idioma. Estes constrangimentos criam um enorme desafio, sendo muito ambicioso criar um modelo independente do idioma. Acontece que a maioria dos autores está focado em documentos em inglês, uma vez que este é o idioma mais explorado e aquele que contém mais dados rotulados. Para obter estes dados são necessários recursos humanos capazes de os classificar à mão. Mais recentemente, as abordagens estão focadas em modelos de Deep Learning que podem levar dias para treinar e consomem milhões de entidades rotuladas.

A minha abordagem é uma abordagem estatística, o que significa que será independente da língua, embora ainda necessite de muito poder de computação. Este modelo combinará múltiplas técnicas tais como Bag of Words, Steeming, e Word2Vec para caracterizar os dados. De seguida, será comparado com dois modelos baseados em transformers, que embora tenham uma arquitectura semelhante, têm diferenças significativas. Os três modelos serão testados em múltiplos conjuntos de dados, cada um com os seus desafios, para conduzir uma pesquisa profunda sobre os pontos fortes e fracos de cada modelo.

Após uma extenso processo de avaliação os três modelos obtiveram métricas superiores a 90% em datasets com grandes quantidades de dados. O maior desafio foram os datasets com menos dados onde o Pipeline obteve métricas superiores aos modelos baseados em transformers.

Palavras-chave: Named Entity Recognition, Processamento de Língua Natural, Inteligência Artificial , Estatística, Independente do Domínio

Contents

List of Figures

List of Tables

Acronyms

BERT Bidirectional Encoder Representations from Transformers ix, 4, 26, 30, 36, 37, 42–54, 56, 57

BNER Biomedical Named Entity Recognition 27

CNN Convolutional Neural Networks 16, 26, 27, 36

CRF Conditional random field 11, 12, 24–28

DBSCAN Density-based spatial clustering of applications with noise 14

DL Deep Learning 16, 19, 20, 26

FDPN Fair Dispersion Point Normalization 9

GRU Gated Recurrent Unit 17, 28

HMM Hidden Markov Models 11, 12, 24

KNN K-Nearest Neighbour x, 13, 25

LSTM Long Short-Term Memory 17, 26, 27

ML Machine Learning 10, 14, 16, 19, 20, 57

NE Named Entity 8, 9, 11, 24, 25, 31, 32, 34, 39, 40, 42, 44, 46, 47, 49–51, 53, 54

NER Named Entity Recognition 1–5, 7, 10, 16, 18, 22–24, 26–28, 57

NLP Natural Language Processing 1, 2, 6, 25, 26, 29, 57, 58

NN Neural Networks 16, 17, 28, 30, 57, 58

POS Part-of-Speech xi, 6, 7, 10, 24, 25

RE Relevant Expression 6, 8–10
RNN Recurrent neural networks 17

SCP Symmetric Conditional Probability 9
SL Supervised Learning 10, 11, 30
SVM Support Vector Machines x, 11, 12, 24, 35

TF-IDF Term frequency-inverse document frequency 8–10

UL Unsupervised Learning x, 14, 15

1 Introduction

Humans are well known for their greed for getting things faster and better. Natural Language Processing (NLP) started has a new desire to improve the relationship between humans and computer information like documents, newspapers, scientific reports, and others. As NLP developed, new sub-tasks appeared, one of them was Named Entity Recognition (NER), or in other words, the ability to detect and extract entities from a document. As examples of named entities we may consider: "United Nations", which is an institution; "New York", being a location; "Barack Obama", which is a person, and 25-12-2021, a data.

With the years passing by, massive development in hardware and communications brought the need for an excellent NER model. Researchers continue their pursuit for state of the art, models mixing innovative techniques with old ones. With this, the big technology companies invest millions of dollars in developing their own systems, for example, Comprehend from Amazon or Watson Natural Language Understanding from IBM. The question is, what is the way to go now that big technology companies have come to the field? NER is still a task in progress, and most of the work done has only been in English. With the advances of the internet, there is a vast increase in non-English data, like Chinese or Korean, and small texts like tweets or Facebook posts, creating ample opportunities to continue the study of NER.

1.1 Context

NER recognizes mentions of rigid designators from a text belonging to predefined semantic types such as Person, Location, Organization, etc. It was first introduced in the Sixth Message Understanding Conference (MUC-6) as a sub-task of Information Extraction, transforming unstructured data from articles to structured information of companies' activities. Since the MUC-6 in 1995, different approaches have been made to develop a model that can precisely recognize entities.

Such as in other artificial intelligence tasks, models needed hardware to achieve results and create the opportunity to explore artificial intelligence tasks at their full potential. In 1995 the internet speed was 28.8Kbps, most people were not connected to the internet, and there were no more than 100,000 websites. Nowadays, we live in a different world. After one crisis and another still going through, everyone in a developed country has access to the internet, where most have multiple social networks, which were invented during this period. The liberty to access the internet through the palm of our hands turned Tech companies the most powerful on earth, and their most significant asset is users' data. As in any other company, social media wants the users to stay connected as long as possible, and for that, they need to feed them with content that they relate to. One way to explore it is to look up the user post and find the aspects they are most interested in. These aspects can be identified by the strong semantics of the named entities, ex: "Black Lives Matter"or "Ukraine". This is an example of the use of NER in the modern world. Besides this, NER is also crucial in other NLP tasks such as:

Information Extration aims to extract data from unstructured textual sources to a more comprehensive machine way through the creation of relationships. A good example may be an input text about the Prime Ministers of Portugal, and the Information Extration creates a sorted table ordered by the year that each one took office. Here, NER can be used to identify names of the prime ministers and dates.

Opinion Mining is used to understand the opinion and emotions of a user towards a product, event, service, etc. In this case, NER identifies the products that a user refers to.

Automatic Text Summarization is a system that selects the essential parts of a document and creates a summary of it. NER is used to find some of the topics in the record and show which parts should appear in the resume.

Machine Translation is a task that intends to translate a text or speech from one language to another. One of the most challenging parts of translation is naming the proper entities to translate since a wrong choice may have a massive impact on the translated sentence's semantics.

Information Retrieval is the problem of dealing with the organization, storage, retrieval, and evaluation of documents due to a query, like the use of web search engines. NER comes in hand to name the entities present in each record for then being classified, sorted, and evaluated by the system.

Question-Answering is a system that generates answers to human questions. NER finds entities in the questions that are then used by the system to look for documents that best correlates with them. After selecting the best documents, it generates an answer for the human, using an automatic text summarization technique.

When comparing the different possibilities for NER is easy to deduct that we have the same problem but in different context. Sometimes there is an extensive *corpus* to analyze, and other times there is just a question. It may be a very formal and professionally written document or an inappropriate comment from a hater in any language of the world. Finally, we may be looking for a Person, Location, or Organization, but it is also possible to look for genes and viruses. Therefore NER can be studied in three dimensions. The language one, which takes into account the different languages that can be evaluated by a model. The *corpus* size. And the type of entities the systems aims to extract [14].

To conclude, the complexity of NER is increasing, and while some authors just make a solution for a specific problem, others create systems that are versatile.

1.2 Motivation

My motivation for this thesis concerns the problems put aside in NER during the most recent years. My primary motivation is to challenge the transformer's architectures in a different context, from English to Swedish, one class to ten, and in different writing styles. While deep learning techniques reported better results, as shown in chapter 2, what is its actual cost concerning hardware and human resources, and how can this affect NER in other languages? The second is to develop a competitive model using statistical and neural network features, combining the engineering power with new state-of-the-art techniques. The third focuses on a multi-language approach. Since most of the work has been done in English, reporting great results, can the other languages, which use a different model, be affected by the lack of exclusivity? The fourth is to explore more types of named entities without losing the quality of the model. Some researchers have already changed the evaluated entities, for instance, in biological or medical documents, besides other fields that are still yet to be explored, like fashion and products. These new entities can be fundamental when applied to short-length text, like questions, conversations, or social media posts.

1.3 Objectives and Contributions

Having in consideration the motivations reported in the previous Section, the initial objectives were achieved and constitute my contribution in this dissertation:

- To create challenging scenarios by changing the language, writing style, number of classes, and corpus size for transformer-based and more classical models.

- To design a model mixing neural networks with statistical features. For this achievement, I developed new features based on the entities' neighbors, entities' stemming, and Word2Vec tested in first point conditions.

- To keep language independence. A statistical features methodology is more language-independent than analyzing a complete sentence in the text. So, efforts were made for the model to be practical in many languages with no changes.
- To develop a transformer model and test it in the same conditions as the first point.
- To use different approaches for the transformers-based model, testing how the techniques succeed and fail in different scenarios.

1.4 Document Structure

The remaining of this document is composed of five chapters that are structured as follows:

Chapter 2 approaches the techniques utilized in NER and how machine learning algorithms extract features from documents to use in their fitting procedures. Then we discuss how authors came along with different paradigms of NER and which solution reported the best results.

Chapter 3 discusses the three different models tested during this research. It starts by presenting a model, fully developed by me, with all features and considerations taken. Then it shows a model based on transformers and how I implemented its neural network. The last one is a model based on Bidirectional Encoder Representations from Transformers (BERT), which can be used in NER tasks with a finetuning technique.

Chapter 4 presents the results of the three methods with seven different datasets. There is a brief explanation of the datasets, followed by the results in three metrics. In the end, a results discussion compares the methods and their performance in the different paradigms.

Chapter 5 is the final one where the conclusions and future works are presented.

2

Related Work

This chapter aims to discuss the different methodologies used to extract entities from the *corpus*. NER can be split into three major phases, Feature Extraction, Model, and Evaluation. All three phases existing since the MUC-6, and there have been numerous developments in all of them. At first, the document presents different techniques or algorithms for the task. The second part gives an overview of the various combinations of methods and models addressed by the researchers and their results.

2.1 Feature Extraction

Feature Extraction is probably the most important of the three phases since we will need good features for the model to classify each entity correctly. We can use different processes to extract information about each entity considering the size of the *corpus*, the words that are nearby the entity, and the own entity. When choosing which technique to apply, we may eventually need to consider the language in the study. There are more than 6000 languages worldwide, and they don't go by the same rules. Let's consider the Japanese language. In Japanese, words aren't split with space or split at all. Spaces are only used after punctuation, which is entirely different from most European languages. This example gives us an idea of how hard it is to make a NER model suitable for many languages. It is possible and highly recommended to combine multiple techniques, creating a vector of features, each one for a particular characteristic, such that the set of features is complete enough to discriminate different entity classes..

2.1.1 Rule Based

Rule-based was the first approach made by the researchers to select named entities. As the name says, it is based on rules to distinguish the different classes. These rules can be case sensitive, like starting with upper case or all the letters are in upper case; suffix or prefix, where checks if a word contains a suffix like *or* for a profession; punctuation, identifying if it is a percentage, maybe if it contains an internal apostrophe represents a person; digit patterns, that are very practical to distinguish dates as four digits represent a year, two

digits with an s a decade. This technique does not fully discriminate entity classes but can work as a first filter feature to eliminate many wrong entity candidates [31].

2.1.2 Gazetteers

Gazetteers is a geographical index or dictionary. In NLP, Gazetteers are also lists of Names, Organizations, and other types of entities besides locations. The authors create these lists by extracting entities from documents, news, or web pages. One way of doing this is by selecting an online news article. On this page, select the HTML tag representing the location of the news. Since every article has multiple links to other news articles, it sets a random new piece and restarts the process. These lists are then used to check if the entity being evaluated is in any list or not. So for each list, there is a position in a binary vector representing if it is contained or not. Although being an efficient tool, these lists are static unless some process regularly updates them [31].

2.1.3 Stemming

Stemming is a technique used to obtain a word stem. It is often implemented as a series of rules applied to a word. It is essential to consider that the final word does not need to exist. It should be the same for all the words it represents. Usually, prefix and suffix are removed, keeping the rest of the word, which most of the time matches the word's root. The actual effect of this technique is to reduce the problem's dimensionality. It is not essential if a verb is in the present or past tense but the relationship between two words. In the following examples, *"Pedro eats an apple"* or *"John is eating an apple"*, it is essential to understand that before the verb *eat* it is a person or animal, not an organization. Table 2.1 shows some examples of the stemming process.

2.1.4 Lemmatization

Lemmatization aims at normalizing each word to its lemma, and the final word can be found in a dictionary. Another difference is that lemmatization outputs the same for words with the same semantic, unlike Stemming. It is crucial in irregular verbs like *to be* or *swim*, where they have different terminations in different terms; or with adjectives like *better* that is transformed to *good*. Lemmatization depends on Part-of-Speech techniques to make a reliable performance and a dictionary that relates each word with its lemma See exemples in Table 2.1.

2.1.5 Part-of-Speech Tagging

Part-of-Speech (POS) is a technique that classifies each word in a sentence to one of the eight morphosyntactic parts of speech, such as noun, pronoun, verb, etc. Identifying the part of speech of a word can help filter RE to discard the wrong ones since RE tends to be

Table 2.1: Comparison of Stemming vs Lemmatization .

Word	Stemming	Lemmatization
Peter	Peter	Peter
is	is	be
eating	eat	eat
an	an	an
apple	appl	apple

noun phrases. English noun phrases have some patterns: Adj-Noun (New York), Noun-Prep-Noun (Bank of England), etc... Although, these patterns are language-dependent since they do not work for Portuguese, for example, where noun phrases usually follow different patterns: Noun-Adj (dias cinzentos), among others. So, Part of Speech tagging can be beneficial but is strongly language-dependent. POS algorithm is composed of a rule-based approach and a HMM. It can benefit NER, but it also has the same difficulties. The semantic ambiguity of words (ex. Orlando, which can refer to a city or a person) imposes a barrier that makes it impossible to be 100% precise. Table 2.2 shows some word tags.

Table 2.2: Part-of-Speech tagging .

John	bought	a	blue	car	from	Mercedes	in	Orlando
NN	VDB	DT	JJ	NN	IN	NN	IN	NN

2.1.6 Bag of Words and Word Embeddings

The Bag of Words concept usually considers a document as a vector of words where the number of word occurrences is represented in each vector position.

However, this methodology can be combined with other approaches to cope with different objectives. Bag of Words can be used to consider the context of an entity. It uses a window of size N, typically N is 5, where the word in the middle is the one being evaluated. Classified entities will have similar bags.

Word embedding is a famous representation of documents recently used as a feature for NER. It captures the context of an entity in a document and creates a vector representation for it. These vectors relate words that are similar semantically or syntactically. Cosine similarity, which measures the similarity between two vectors by inner product, is used to measure the similarity between the embeddings. It outputs a continuous value between 0 and 1, whereas close to 1, the more similar the vectors are. Different models have different approaches to creating a word embedding, a famous one in the Word2Vec with CBOW, which uses a bag of words with dimension C to feed the algorithm. The Figure 2.1 shows which words are semantic close to *car*.

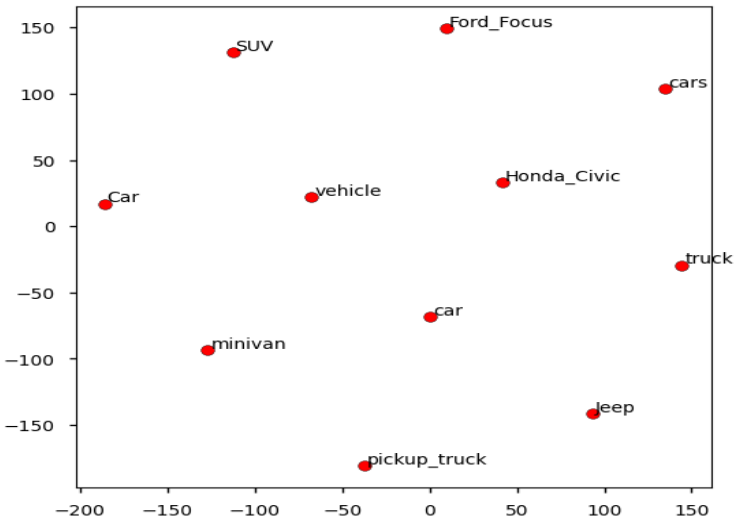

Figure 2.1: Most similar words to *car* using Word2Vec

2.1.7 Document and *corpus* features

Document and *corpus* features are defined by both content and document structure. Instead of focusing only on one word at a time, we take advantage of the *corpus* to find out new features about the statistics of an entity. Some valuable metrics to consider are the number of times a word appears in a document (term frequency), the number of documents a word seems (document frequency), the subject of a document, co-occurrences.

All this information can help to understand from the starting point which words or sequences of words are entities or not. Let us look at the single word *the*. It is the most used word in English, and it will never be a NE since it is semantically weak. Like *the*, there are other examples like *a*, *be*, etc. that are known as stop words. However, a considerable number of RE are semantically strong units, such as *Bank of England* or *Primeiro Ministro da Nova Zelândia*, which are good candidates to be Named Entity (NE), while containing stop words within it. The nature and the specificity of the *corpus* can be crucial to explore new entity types.

2.1.8 TF-IDF

Term frequency-inverse document frequency (TF-IDF) is a well-known metric to represent how important a word is to a document in a collection of documents. Term frequency, as explained before, is the number of times a term i appears in a document j ($tf_{i,j}$); and document frequency (df_i) is the number of documents a word i appears. This metric is often used in search engines to qualify pages considering the user queries. In formula 2.1 N stands for the number of documents in the *corpus*.

$$Tf - IDF_{i,j} = tf_{i,j} * \log(\frac{N}{df_i}) \tag{2.1}$$

This metric can select the meaningful words from documents, such as *China*, *Obama*, *Putin*, *Portugal*, *lungs*, etc., which are clearly NE due to their strong semantics. So, TF-IDF can be a tool to contribute to select NE.

2.1.9 Extraction of Multi-grams

Three tools working together, are used for extracting RE from any *corpus*: the LocalMaxs algorithm, Symmetric Conditional Probability (SCP) statistical measure and the Fair Dispersion Point Normalization (FDPN).

Thus, let us consider that an N-gram is a sequence of words in text. For example the word *Minister* is an 1-gram; the sequence *Prime Minister of Portugal* is a 4-gram. LocalMaxs is based on the idea that each N-gram has a kind of "glue"or cohesion sticking the words together within the N-gram. Then it is possible to distinct the N-grams quality according to their cohesion metric. One can intuitively accept that there is a strong cohesion within the N-gram between the words *Berlin* and *Wall* making the *bi*-gram (*Berlin* , *Wall*). However, there is not a strong cohesion within the N-gram (*to* , *the*). To calculate the cohesion in a N-gram the authors [18] use *SCP* or other metrics. So, the SCP(.) cohesion value of a generic bigram (x,y) is obtained by Equation (2.2)

$$SCP((x,y)) = p(x|y).p(y|x) = \frac{p(x,y)^2}{p(x).p(y)} \tag{2.2}$$

where $p(x,y)$, $P(x)$ and $p(y)$ are the probabilities of occurrence of bigram (x,y) and unigrams x and y in the *corpus*; $p(x|y)$ stands for the conditional probability of occurrence of x in the first (left) position of a bigram, given that y appears in the second (right) position of the same bigram. Similarly $p(y|x)$ stands for the probability of occurrence of y in the second (right) position of a bigram, given that x appears in the first (left) position of the same bigram.

However, in order to measure the cohesion value of each N-gram of any size in the *corpus*, the FDPN concept is applied to the SCP(.) measure and a new cohesion measure, SCP_f(.) , is obtained.

$$SCP\_f((W_1...W_n)) = \frac{p(W_1...W_n)^2}{F} \tag{2.3}$$

where

$$F = \frac{1}{n-1} \sum_{i=1}^{i=n-1} p(W_1...W_i).p(W_{i+1}...W_n) \tag{2.4}$$

where is the probability of the N-gram $W_1...W_n$ in the *corpus*. So, any N-gram of any length is "transformed"into a pseudo-bigram that reflects the average cohesion between each two adjacent contiguous sub N-grams of the original N-gram.

Then, LocalMaxs algorithm elects the cohesion of the N-gram is greater or equal than the cohesion of any of its contiguous sub- $n-1$-grams, and greater than the cohesion of any $n+1$-gram of which it is a sub-sequence of words. Analyzing the following N-gram

The King of Sweden visited it is easy to deduct that *King of Sweden* should be the extracted RE, but it had to compare it with the other $n+1$-grams (*The King of Sweden* and *King of Sweden visited*) and $n-1$-grams (*King of* and *of Sweden*) to be select as a RE .

2.1.10 Tokenization

Tokenization is applying a function that transforms words into tokens. This function may change according to the objective. Using the NLTK, it splits a sentence by his words [35]. Another famous tokenization function is the BIESO. *B* represents the beginning of an N-gram; *I* the inside of an N-gram; *E* the end of an N-gram, *S* represents a unigram; and *O* represents an Outsiders, in other words, an entity that it is not a named entity. Applying BIESO will distinguish the N-gram in our data.

2.1.11 Conclusion

To conclude, this Section is important to reflect how each technique is affected by the different dimensionalities of NER.

Analyzing the Language Dimensionality, Rule-Based, Stemming, Lemmatization, and Part-of-Speech are all language-dependent, so a system can not be multilingual when developed with some of these tools. The best that can be done is using different tools in different languages, but it is not easy to find it for other languages besides the most spoken ones. Bag of Words, TF-IDF, Document, and *corpus* features are not language-dependent and can be easily transferred from one language to another.

In terms of *corpus* size dependence, the quality of the results obtained by TF-IDF and the Extraction of Multi-grams heavily depend on big *corpus*. Without them, these tools may become useless, providing a misleading output. On the other side of the balance, the remaining techniques are not *corpus* size-dependent.

Regarding the sorted entities, Rule-Based, Stemming, Lemmatization are dependent on our target entities. So if the model is made to be more flexible concerning the extracted entities, corpus features and TF-IDF are the suitable options.

2.2 Learning methods

As in other Machine Learning (ML) problems, we have different approaches to classify the entities and as always there are some constraints that make us choose one method over the other, such as the data we have, if it's annotated or not, and the cost to annotate it.

2.2.1 Supervised Learning

Supervised Learning (SL) is a methodology in ML that utilizes labeled data to train a model that predicts an output. When the output is a class, such as in Named Entity

Recognition applications, it is called Classification. When the output is a continuous value, it is called Regression, which is not in the scope of this thesis. In our case, the class that we want to indicate is the type of Named Entity, and the input is a vector of features. The quality of the features is crucial to provide a good clustering division, that is, no overlap between clusters. The model needs to train, or in order words, to find out relations and patterns between the input and the output to distinguish the different classes. The second step is to test the model, which is done by applying unseen data to the model and comparing the handmade label with the models' label. The test step is critical to check if the model is overfitting, adapting too much to the training data, or underfitting. The objective is to find the sweet spot between overfitting and underfitting by fine-tuning the models' parameters and training with different data. Unfortunately, SL depends on labeled data, which requires many hours and human resources, and time to train, which may go from a few hours to a few days [31].

2.2.1.1 Decision Trees

As the name suggests, it can be visualized as an inverted tree where a node is a decision based on the data features, and a leaf is a classification for the input. Since it is a greedy algorithm, the tree's base will have the features that cost the least; in other words, the element that best splits the data.

2.2.1.2 Support Vector Machines

Support Vector Machines (SVM) finds a Hyperplane in an N-dimensional space, N being the number of features. SVM selects the two closest points to the possible hyperplane, which are called support vectors. Since it is possible to find multiple hyperplanes capable of distinct the data, SVM maximizes the distance between the support vectors and the hyperplane. This approach reinforces the confidence that future vectors will be better classified, decreasing the overfitting of the algorithm. After finding the best hyperplane, a vector is classified by the hyperplane's side. To deal with outliers, SVM softens the margins of the hyperplane. It is done by allowing some vectors to go through the hyperplane and not be considered as the support points; this allowance is tunned by hyperparameters. Even if it misclassifies the vector, it generally creates a better classification line. This concept is called soft margins and can be observed in Figure 2.2 . Sometimes there may be no hyperplane able to distinguish the classes. In these cases, SVM applies a kernel trick consisting of mathematical functions, like a third-degree polynomial, to the features, making the data linearly separable. The Figure 2.3 represents it.

2.2.1.3 Hidden Markov Models, Conditional random field and Viterbi Algorithm

Hidden Markov Models (HMM) is a powerful statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence. This algorithm is based on the Markov Process, a series of states that depend

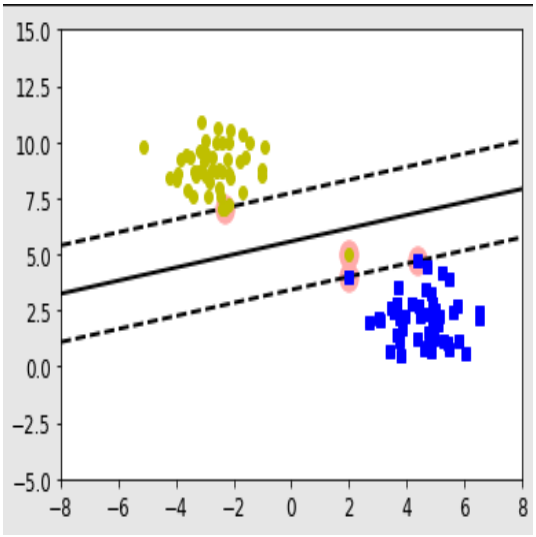

Figure 2.2: SVM using soft Margins

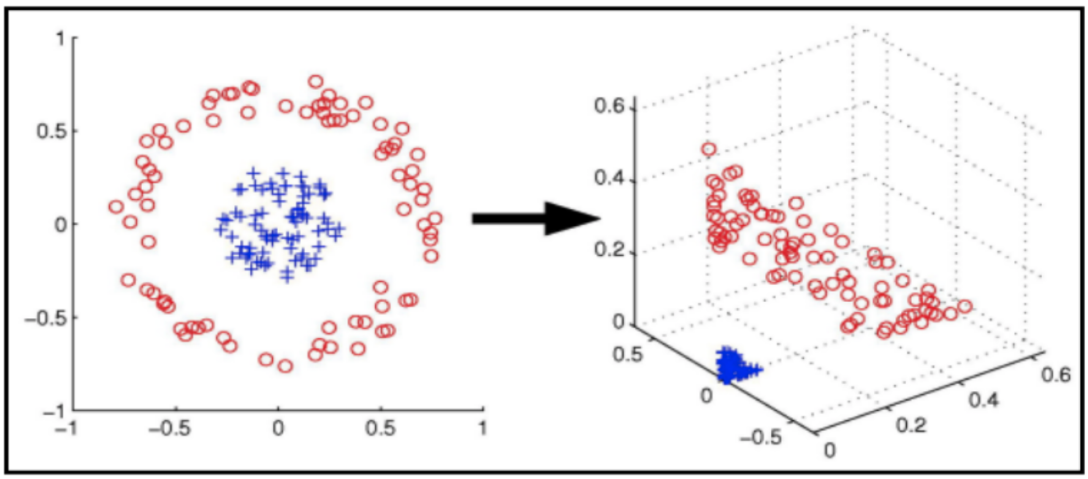

Figure 2.3: Applying a Kernel trick in SVM

only on the previous one. It's called hidden because the underlying process is not visible to the outside. [5]

Conditional random field (CRF) is a discriminant classifier that takes into consideration the context of the input. Although it is similar to HMM due to both being used to deal with sequential data, CRF models the conditional probability distribution. It also does not rely on independent data. In fact, HMM is a particular case of CRF.

Viterbi Algorithm is not a Supervised Learning model but a dynamical programming algorithm that allows us to compute the most probable path considering the input and the model.

2.2.1.4 K-Nearest Neighbour

The K-Nearest Neighbour (KNN) algorithm assumes that similar things exist nearby. It aims to locate all of the closest neighbors around a new unknown data point to figure out what class it belongs to. It is a distance-based approach. K is defined in the algorithm's hyperparameters and represents the number of neighbors a new point will consider to find out his class. So, when a new point is added, it calculates the distance to each point and selects the closest K points to him. Then, the new point classification will be the same as most of the points K closer to him. In cases where there is no majority, the class will be selected as odd. A critical phase is to fine-tune the K value. A smaller K creates an overfitted model, capturing noise data and failing to perform with test data. If a larger K value is selected, the model will underfit the data and fail to perform well in the training data. This can be observerd in figure 2.4

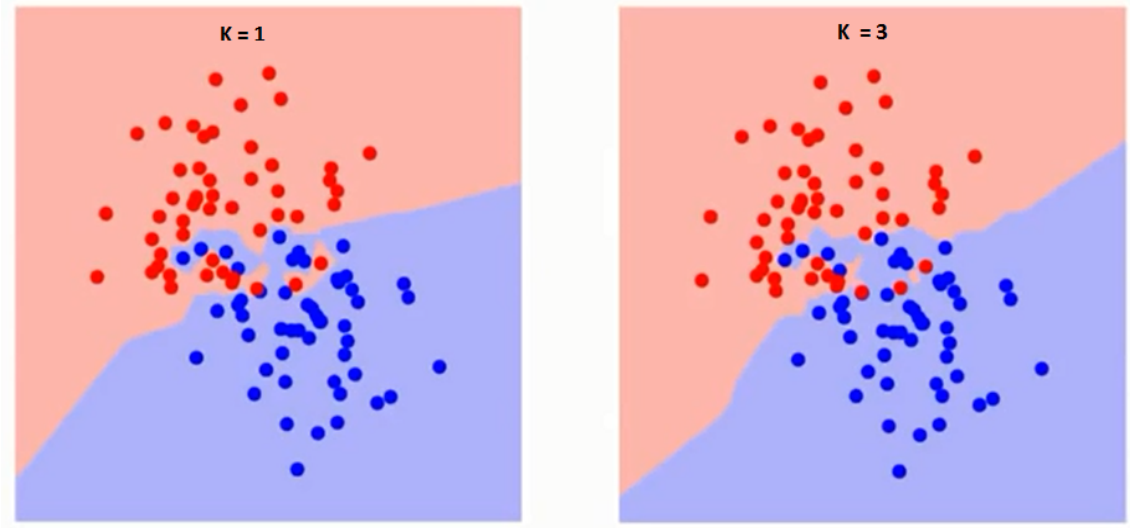

Figure 2.4: KNN classification of the same data with different K

2.2.1.5 Naive Bayes Classifier

Naive Bayes Classifier is based on the Bayes Theorem, where it gives the probability of A happening knowing that B has already occurred. It also assumes that features are independent of each other. In other words, the occurrence of one feature does not affect the others, which is why it is called naive. Another important note is that all features have an equal effect on the classification. The Bayes algorithm has the following expression

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.5}$$

where Y represents the output class and X the features. So, X can be seen as vector of n features, such as $X = (x_1, x_2, x_3, ..., x_n)$. Then it is possible to substitute in the Bayes Theorem for:

$$P(Y|x_1,,...,x_n) = \frac{P(x_1|Y)P(x_2|Y)P(x_3|Y)...P(x_n|Y)P(Y)}{P(x_1)P(x_2)P(x_3)...P(x_n)} \tag{2.6}$$

For all data, the denominator does not change, it remain static. So, it can be seen as a proportion.

$$P(Y|x_1,...,x_n) \propto P(Y)\prod_{i=1}^{n} P(x_i|Y) \tag{2.7}$$

The class of the input will be the one with higher probability according to the features

$$\hat{Y} = argmax_Y P(Y)\prod_{i=1}^{n} P(x_i|Y) \tag{2.8}$$

2.2.2 Unsupervised Learning

Contrary to Supervised Learning, Unsupervised Learning (UL) does not use annotated data to train its model. This type of algorithm finds out the patterns in the input data to cluster it. The typical output is clustering, and the optimal cluster number may not be the same as the real number of classes of entities we are trying to predict. We can use different algorithms in UL like Spectral Clustering, DBSCAN, etc. [31]

2.2.2.1 Clustering Algorithms

Density-based spatial clustering of applications with noise (DBSCAN) is a well-known clustering algorithm used in ML that groups points that are closed to each other through a distance metric, usually Euclidian distance, and a minimum number of points. This algorithm needs two inputs:*eps*, which specifies how close points should be to each other to be considered a part of a cluster; and *minPoints*, the minimum number of points to form a dense region. Points that are in low-density spaces are classified as outliers [42]. This algorithm is suitable to locate clusters having irregular shapes.

K-**Means** cluster the points by their distance between each other. Its starts with *K* centroids (an input parameter), where each point is associated with the closest centroid. Then it calculates a new position for the centroid, considering the points that have selected it as the closest one. This iteration of defining a new position for the centroid keeps going until the centroid position stabilizes or we get to the maximum number of iterations defined by the user [19]. Due to the euclidean distance usually used, this clustering method assumes that clusters are spheroid and all with the same volume, which is not the case in most situations. The clustering quality depends on the number of centroids defined by the user.

Spectral Clustering is one of the best unsupervised learning algorithms. It has three phases. At first, it builds a similarity graph between all points; in this case, similarity will be the distance between the points. Secondly, it projects the data in a lower-dimensional space computing the graph laplacian matrix. Then it calculates the Degree matrix, which is a diagonal matrix with the degree of each point. Then the graph Laplacian matrix

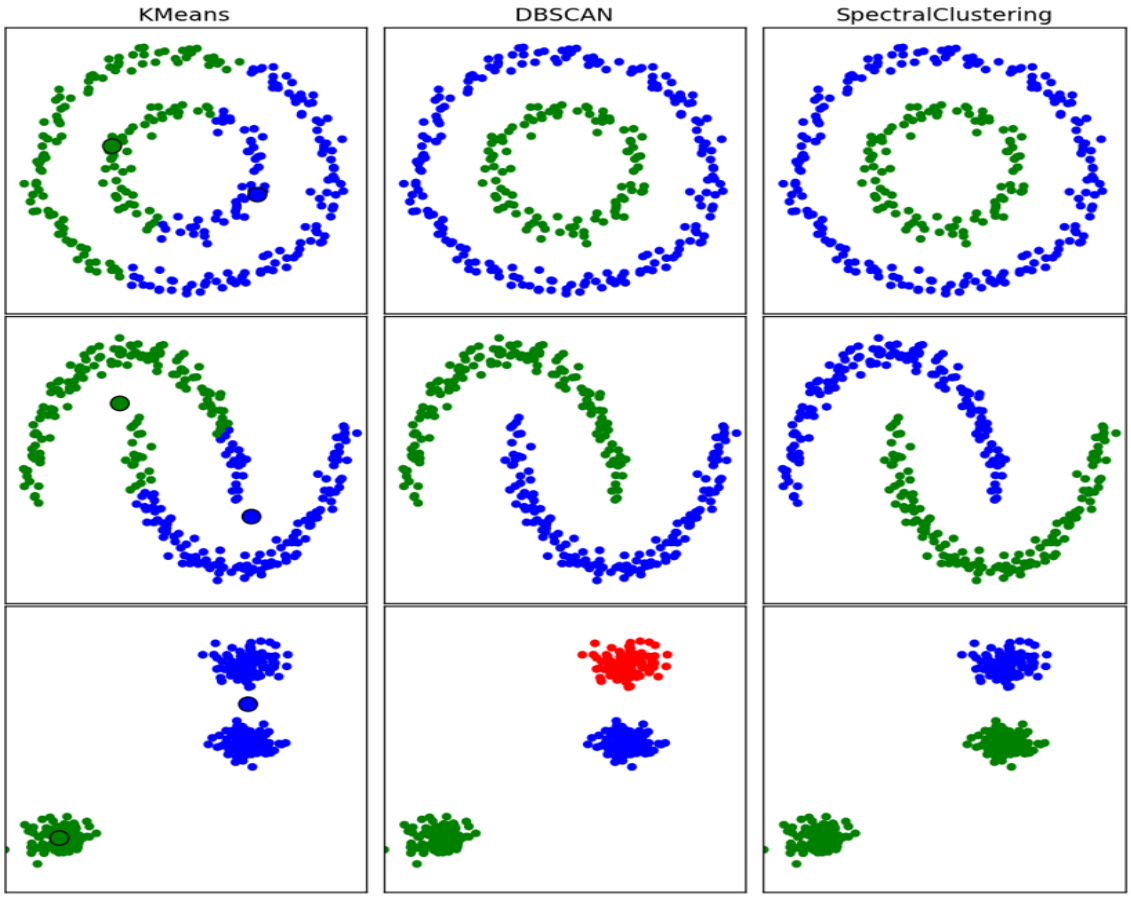

Figure 2.5: Comparison of Different UL Algorithms in different data sets

is equal to the Degree matrix minus the similarity matrix. The last step is to cluster the reduced data using a traditional clustering algorithm like K-Means. This algorithm clusters data that may not be close together, but it is obviously in the same cluster (like in a ring); it is fast and easy to implement but does not scale well with data size [30].

Figure 2.5 provides a visual comparison between the studied algorithms in different data sets.

2.2.2.2 Measuring the Quality of the Cluster

An important note in some algorithms is that they may require selecting the number of clusters. In that case, the user needs to calculate *a posteriori* the more likely number of clusters by measuring the clustering quality of each clustering proposal, using well-known algorithms like Silhouette score or Sum of Square Errors.

Silhouette score is a technique to calculate the quality from the clusters. Its values range from 1 to -1, where 1 is an optimal clustering and values close to -1 means that clustering was poorly done. To calculate, we need A, the average distance between the points in a cluster, and B, the distance between the points at different clusters. With this,

we just need to:

$$SilhouetteScore = \frac{B - A}{max(A, B)} \tag{2.9}$$

Sum of Square Errors is the sum of the squared distance between each point and the cluster mean. Objectively it measures the variation in a cluster.

2.2.3 Semi-supervised Learning

Semi-Supervised Learning is a technique that mixes Supervised Learning with Unsupervised Learning. It uses a small amount of annotated data with a big amount of unlabeled data to train the model. In this case, the authors [39] wanted to create a list of Location named entities. Although it would be simple to create a list with the countries and their capitals, the same do not apply to cities. So, the authors created a model that recognizes new cities through patterns. To create this patterns a seed must be given to the model, in this case the seed is a city name. Then the model finds the pattern according to the seeds and looks for others entities that follow the same pattern.

2.2.4 Deep learning

Deep Learning (DL) has been state-of-the-art for recent years in most tasks from Natural Language Processing. The artificial neurons compose layers of artificial neural networks inspired by the brain's structure. The first layer is the input one, and the last is the output layer. The layers from the middle are not "visible", and from this fact comes the name Deep. These layers are composed of nodes that receive input, and based on what has previously been learned, it computes an output that will be the input of the next layer. The essential advantage of deep learning is representation learning and the semantic composition empowered by both vector representation and neural processing. This allows a machine to be fed with raw data and to automatically discover latent representations and processes needed for classification or detection [22]. There are three critical facts for DL thriving in NER: first, it benefits from non-linear transformations that cannot be done in every classical ML model; second, deep learning is, say, more independent from the features; and finally, it is easier to make complex systems.

2.2.4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is comprised of neurons that self-optimize through learning like Artificial Neural Networks. The significant difference is that CNN is primarily used to find patterns in data, usually imaged focus data, due to their three-dimensional structure. The first layer is the convolutional layer, which filters the input data. The input may be a word or a sentence, depending on the application of the Neural Networks (NN). This filter is applied to all the data to detect unknown patterns used to classify the input. Convolutional layers are commonly referred to as partially connected layers since the output is not directly connected to the input but either with the result of the

filter application. Another Convolutional layer can follow a Convolutional one, creating a hierarchical pattern detecting structure, where earlier layers detect simple patterns, and later ones detect the combination of these patterns. The pooling layer filters the input data applying an aggregation function. This layer reduces the dimensionality of the data while preventing the overfitting of the model. The most used aggregation function is the max pool, which selects the most valuable parts of the data while moving to throw it, but there is also the average pool, which makes an average of the filtered data. The final layer is the Fully-Connected layers, where each node is connected to all the nodes in the previous layer. This layer is responsible for classifying the data, using a softmax activation function, predicting a value between 0 and 1 for each class.

2.2.4.2 Recurrent neural networks

Recurrent neural networks (RNN) are widely used for modeling sequential data. RNN are known for their memory since it uses information from older inputs in the current input. RNN interprets inputs as sequential instead of independent data like other NN models, allowing NLP models to connect words and identify N-grams. Another difference is that RNN layers share the same weighted parameters within each layer, which are still tunned in with backpropagation and gradient descent. Since it implements backpropagation using gradient-based learning methods, it suffers from the Vanishing gradient problem. When we train a model, the gradient, which measures the change in all weights regarding the error, will be so slight that it may stop training.

Long Short-Term Memory (LSTM) is an advanced RNN that minimizes the Vanishing gradient problem using the information from events that happened thousands or hundreds of times steps earlier.

Gated Recurrent Unit (GRU)) is a variation of RNN that solves the Vanishing gradient problem and is a big rival to LSTM [16].

2.2.4.3 Deep Transformers

Deep transformers like RNN captures the timely dependencies in sequence, but it adopts a different mechanism for this purpose. Transformers use self-attention, which gives other importance to distinct parts of data. Although each word has sequence data, it does not need to be processed by order. It is possible to paralyze the training phase, making it much faster than RNN models. The transformer encoder starts with adding the positional encoder to the word embedding. This prevents the model from losing the sequence of the words. Then comes the encoder block, composed of:

- Multi-Head Attention layer, which produces the attention vectors of each word. It is a Multi-Head because each word produces multiple attention vectors that will produce the final vector through a weighted average. An attention vector contains the relationship between the word it represents and the words in the input. For each relation, the value range from 0 to 1, where 1 represents a stronger relationship.

- Feed Forward is a simple layer applied to every attention vector to transform the vector to a more comprehensive structure for the next encoder or decoder. Each attention vector is passed one at a time, but it is possible to paralyze the process with multiple Feed-Forward Networks layers since each vector is independent.

The transformer decoder starts the same way as the encoder, with an output embedding and a positional encoder. Then, it is composed of three blocks:

- Masked Multi-Head Attention, which makes the attention vectors from the output embedding. It is called masked because it can only represent relationship between the word it represents and the previous words.
- Multi-Head Attention, which takes the vectors from the encoder and the Masked Multi-Head. It produces another attention vector representing the relationship between both input vectors.
- Feed Forward makes the attention vector more comprehensive for the Linear layer.

In the end, a linear layer and a Softmax algorithm will calculate the probability of each output class according to the input. The class with a higher probability will be the label of the input word. The described components can be seen in the figure 2.6.

While the transformers architecture was designed to a sequence-to-sequence task, it can be adapted to fit in different categories. These changes are easily done my using only part of the architecture creating three types:

- The Encoder-only which converts a sentence to a numerical representation that can be used in text classification task such as NER.
- The Decoder-only which given a sentence tries to predict the next word.
- The Encoder-Decoder which is used in translation or summarization tasks converting one sequence of text to another.

2.2.4.4 Deep Learning Techniques

In addition to the presented network architectures, a few more techniques are being explored in NER that are presented next.

Deep multi-task is a technique that learns different tasks in just one model. Since it has multiple tasks, it needs more training data, but it will be more heterogeneous to learn more generalized features. An additional advantage of this model is learning a feature that can be easily identified in task A but can also be applied to task B. Considering these advantages, it is easy to understand that these models can not suffer from overfitting.

Deep Transfer Learning is a technique that mimics the human brain's ability to use the knowledge learned from one situation in the others. A real-life example is when learning a new language, and it is possible to know the meaning of a word just from

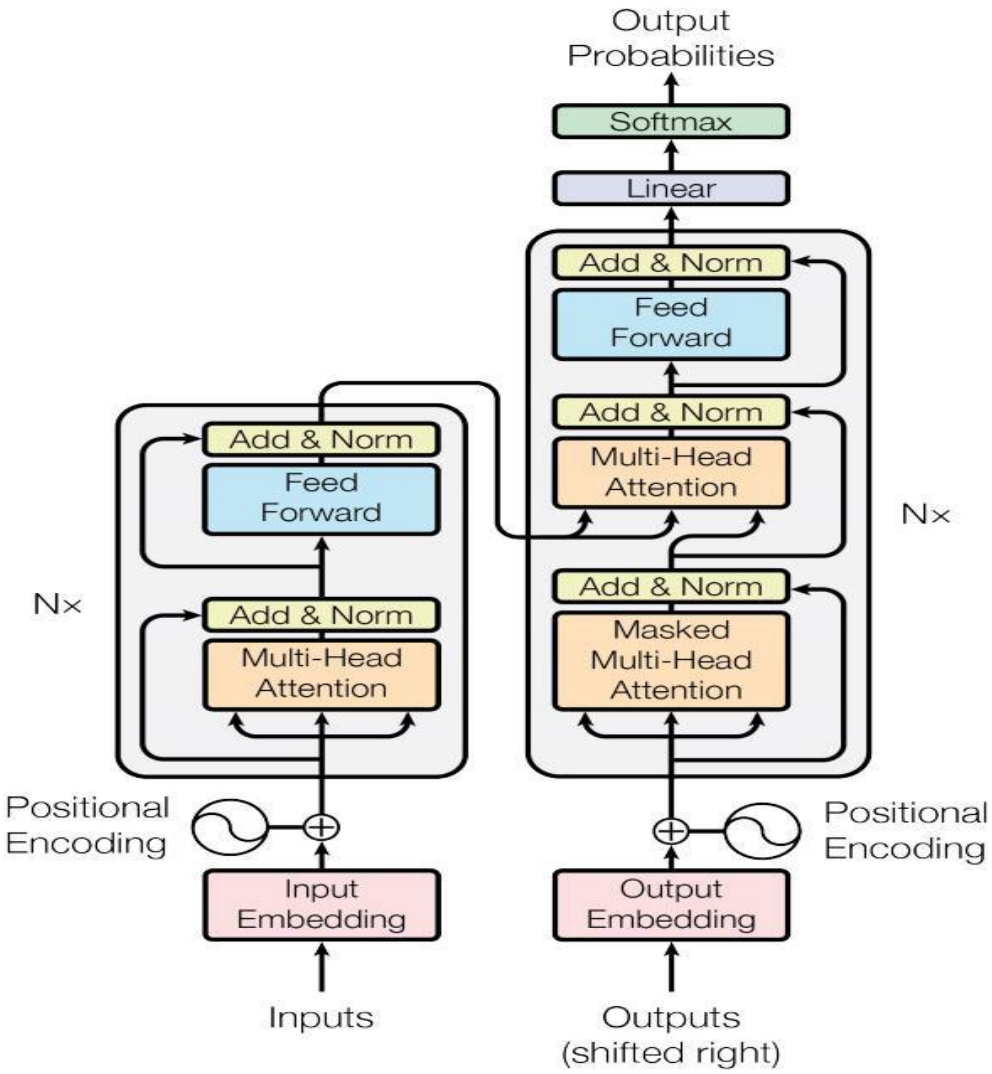

Figure 2.6: Transformer encoder at the left and a Transformer decoder at the right

the similarity between it and the learner's mother language. In Deep Learning, this is very useful since some models take more than one day to finish the learning process, and applying this step to every task would have a considerable cost. So Deep Transfer learning pre-trains a model that is then retrained with new data for a specific purpose. This is also very helpful in tasks that do not have a large data set, whereas the model is pre-trained with generic data and finned with specific data.

Deep Active Learning is a mixture of DL with Active Learning. Active Learning is a method that assumes that each sample of data can be more or less helpful for training the algorithm. In other words, it selects the data that is more beneficial for training the model, reducing the cost of obtaining the labeled data and training the model. On the other hand, DL is a very robust ML model that is highly dependent on data to be prepared. The marriage of these two fields is expected to achieve superior results while reducing

the dependence on labeled data [38].

Deep Reinforcement Learning is another union of two fields in ML, DL with Reinforcement Learning. It is inspired by the human behavior of trial and error, creating a new way for the models to learn the data. Reinforcement Learning can be seen as a player in a game where each good decision is rewarded, and a bad one is penalized. DL helps Reinforcement Learning to solve new problems that at first were unsolvable due to the high-dimensional state and action spaces [2].

Neural Attention is the ability to give a different weight to different parts of the data. This is used in Transformers but can also be applied when analyzing a character or word-level feature.

2.2.4.5 Activation Functions

An Activation function is responsible for deciding whether or not a neuron is activated. It takes the input from the previous layer and decides the output for the next until it reaches the end of the neural network. Having an Activation function is more important than it may look. The neural network would consist of linear transformations that would always have the same behavior if it did not exist. The image 2.7 have the graphical representation of Sigmoid, Tanh and ReLu functions.

The Rectified Linear Activation Function is a nonlinear function that acts like a linear one, allowing for complex data relationships to be learned. If the input is positive, it returns the same value as the input. If the input is negative, it returns zero. Due to its similarities with linear functions, it preserves many properties that make linear models generalize well [33].

The Sigmoid function has an S shape that outputs between zero and one for any given number. The bigger the input, the closer the result is to one; the lower is, the input has closer to zero its outputs [34].

The Tanh function is also an S shape but ranges from -1 to 1. It is different from the Sigmoid function as it will map the negative inputs strongly negative and will map the zero inputs near zero [37].

The Softmax activation function converts a vector of K real numbers into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. Due to this, it is used as a final layer in tasks with more than two classes. The outputs will vary between 0 and 1, and the sum of all is equal to 1. Softmax follows the following equation for $i = 1...K$ and $z = (z_1, ..., z_k) \in \mathbb{R}^K$ [6].

$$\sigma(z)_i = -\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.10}$$

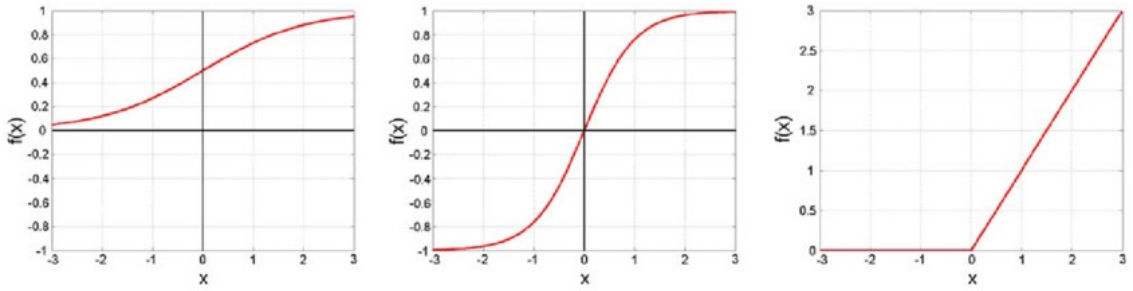

Figure 2.7: Graphical representation of Sigmoid, Tanh and Relu activation functions

2.2.4.6 Loss Functions

The loss function is responsible for qualifying how a model learns his task. It uses the input class and model output as input, ranging from close to zero to infinity, where close to zero means it learned well, while the higher the number worst the model performance. The neural network will update itself during training to minimize the loss function. In the case of NER, which is a single label classification task, the most common loss functions are:

Categorical Crossentropy loss function calculates the loss using the following equation, where Y_i is the target value and $\hat{Y}_i$ is the predicted value.

$$Loss = -\sum Y_i \cdot \log(\hat{Y}_i) \tag{2.11}$$

This loss is an excellent measure of distinguishable two discrete probability distributions from each other. In this context, Y_i is the probability that event i occurs and the sum of all Y_i is 1, meaning that precisely one event may occur. The minus sign ensures that the loss gets smaller when the distributions get closer. It is highly recommended for this function to be used after a Softmax activation function [13].

Focal loss is an extension of the Categorical Crossentropy. A weighting factor is added to the function that allows the focal loss function to focus on actual mistakes. The Categorical Crossentropy gives a much higher loss value than the Focal when the probability of ground truth class gets lower.

$$Loss = -\sum (1 - \hat{Y}_i)^{\gamma} \cdot Y_i \cdot \log(\hat{Y}_i) \tag{2.12}$$

Once again Y_i is the target value and $\hat{Y}_i$ is the predicted value. the γ is a parameter that adjust the rate that the examples are down-weighted [23].

2.2.5 Conclusion

Summing up, all the algorithms presented are not language or entity-dependent. Besides that, Deep Learning algorithms are *corpus* dependent since they only provide good scores when trained by millions of documents. Even though it is possible to bypass this problem

with learning transfer, it still needs a cluster to train the model for more than one day. When choosing a suitable model, this back-and-forth between document size and resource capability is the only consideration.

2.3 Evaluation

Evaluation is the last part of the NER process. It is helpful for the researcher to get an overview of the model's performance and compare it with the other models already developed. To determine if an entity is incorrect, the output is compared with the solution, which must be labeled by hand. But before going deeper into the different evaluation models, it is essential to understand the 5 mistakes a model can make.

1. The model identifies an entity where there is none
2. An entity was missed by the model
3. An entity was poorly classified by the model
4. An entity was well classified but got its boundaries wrong, such as identifying *York* instead of *New York*
5. An entity was poorly classified and got its boundaries wrong

The most used metrics for evaluating the performance in classification context, are the Precision (Eq.2.13), the Recall (Eq. 2.14) and F-Score (eq.2.15).

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.13}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.14}$$

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.15}$$

2.3.1 MUC evaluations

Message Understanding Conference (MUC) evaluation, developed for MUC events, split the classification of the type and text in two. So if a model gets the right type without getting the correct text, it receives a point and vice versa. This is a softer evaluation metric since it will give better performance even when a model makes a mistake. In the following sentence,"I Love New York", if the model classified New York as an entity and a location, it would get 2 points. If it ranked New York as a person, it would get 1 point for the text, and if instead of identifying New York, it marks just York as a location, it would get just 1 point for type. To calculate the Precision of the model, it divides the number of points for the number of entities * 2. For the Recall, we divide the number of points

for the number of correct entities * 2. The measure used to compare the systems is the F-Score, a harmonic mean of Precision and Recall.

2.3.2 Exact-match evaluation

Exact-match evaluation is a simple protocol that classifies as correct if an entity has the exact type and text. All the other cases are classified as mistakes. The models using these systems compare to each other using the micro-averaged F-measure. It needs a micro-averaged Precision and a micro-average Recall. The micro-averaged Precision divides the sum of *true positives* in each class and divides it by the sum of positives in each class. The micro-average Recall divides the sum of *true positives* in each class and divides by the sum of the number of entities by class. The equations 2.16 and 2.17 are the micro-average Precision and micro average Recall

$$micro - averagePrecision = \frac{TP_0 + ... + TP_n}{TP_0 + ... + TP_n + FP_0 + ... + FP_n} \tag{2.16}$$

$$micro - averageRecall = \frac{TP_0 + ... + TP_n}{TP_0 + ... + TP_n + FN_0 + ... + FN_n} \tag{2.17}$$

where n in the number of classes.

Another important metric is the macro, which can be done to Precision, Recall and F-Score. The macro works as a arithmetic average, which sums all the values and divides by the number of classes.

2.3.3 Automatic content extraction evaluation

Automatic content extraction aims to automatically infer from human language data the entities being mentioned, the relations among these entities that are directly expressed, and the events in which these entities participate [11]. Even if it is a more complex problem than NER, its evaluation is different from the ones we have mentioned before. Automatic content extraction evaluation gives different weights to each type of entity, meaning that having one person entity correctly identified is not the same as having a location one. The system starts with a 100% classification that is then penalized for each mistake done by himself.

2.4 State of Art

After an extensive revision of the most used algorithms and techniques used in NER, this Section will discuss how they were combined by different authors. It is essential to consider that while some models are focusing on solving NER in English with just three classes, others may have a large number of classes. Language is also a vital aspect to consider since most of the older implementations were only focused on English due to the labeled data. Nowadays, more languages are taken in regard and models made

for multi-languages. The last aspect is the data domain and if the model is made for multi-domain or not.

D. Bikel et al. made a model based on HMM for English and Spanish. They considered words a bi-dimensional tuple, where the first position was the word and the second a word feature. These features were ordered, and when a word was meaningfully reflected in more than one features, the one with a higher rank is selected. The features varied from containing a certain number of digits, having a dot or a slash, starting with a capital letter, or being the first word in a sentence. In total, there were 9 features. Since it implemented HMM model, it analyzed words as bigrams, they came up with a solution to deal with unknown bigrams that were to train a separate model to deal with it. The model was trained in English with MUC-6 data set and in Spanish with articles from the agency AFP. It reported 93% for English and 90% for Spanish with the F-Measure metric using an exact match evaluation technique. [4].

A. McCallum and Wei Li implemented a CRF model for English and German that makes an efficient feature induction. This model starts with no feature, and after a few rounds, it selects the features that most increase the log-likelihood of the correct path and trains the weight for the new features. This model that at the end contained 1,000 features reported an 84.04% F-Measure in English and 68.71% in German [27].

Asahara, Masayuki, and Matsumoto made an SVM model for Japanese where besides the usual character features extracted, they developed a unique solution using POS and Chunking. They applied a POS tagger that retrieved with n-best answers for each word (in Japanese, a word is a character), then the n-best results with some other more usual features are annotated for each word fed into the SVM model. After some tests, they conclude that POS and character level features are crucial for NER in Japanese, where it got an 87.21% in F-Measure, and in other languages that have word unit problem [3].

D. Nadeau et al. proposed an unsupervised algorithm that combines a Named Entity Extraction with a Named Entity disambiguation model. The Named Entity Extraction model creates gazetteers (named-entity lists) from the web. It starts with a query of 4 named Locations (this can be applied to any other entity) called seeds. Each retrieved HTML page uses a Supervised Web Wrapper algorithm to extract similar cases as in the input Query. They repeat this process with new seeds from the gazetteer that are not noisy if possible. In the Named Entity disambiguation model they used, three problems were identified:

- Entity-Noun Ambiguity. It occurs when an entity is a homograph and their solution, which follows the heuristic proposed by Mikheev [28], assumes a word is a Named Entity unless in three cases: It is at the start of a sentence; appears at a document without the initial letter as a capital; and if it is in a sentence where all words are in capital form.

- Entity Boundary Detection. Here they use the principle of the longest match, merging all consecutive entities of the same type and entities of other types with a capital

letter.

- Entity-Entity Ambiguity. It happens when an entity has more than one possible classification. To solve that, they made a simple alias resolution algorithm that when an entity is unclear, it classifies as the common entity to both words. For example, for "Atlantic ocean"where *Atlantic* can be a Location or an Organization, it results in the Location type.

In the end, Nadeau evaluated his model with the MUC-7 data set, performing a 73% in *F*-Measure. It was only tested in English, but he considers this algorithm can be applied to more languages and more than 3 entity types. [32].

J. Heng and Grishman suggested a Supervised NE algorithm improved by combining bootstrapping with self-learning. Bootstrapping trains a segment of the data tests. The new model rejects the data sets that did not improve, and finally restarts the process with the new improved model. The self-training model creates an order by which the documents are learned. It prioritizes the documents containing the entities with higher confidence merged with the previous training data to start a new iteration on labeling. Then they studied the impact of each document on the performance of the model. They come to the fact that having a large data set is not enough by itself. There must be a systematic process of selecting the richer documents on the quality of entities. Their model reported 89.1% in *F*-Measure for English and 90% in Chinese [17].

Twitter is the only Big Social Network with an open API for researchers to use their data in various studies. In NLP, tweets are widely explored since their unique characteristics. A Tweet is a small message of 140 characters containing tags to other users, images, links, or emojis. These characteristics make them quite different from the standard *corpus* studied. The small length of the text with the informal writing and a higher chance of an orthographic error make the use of classical POS taggers unreliable and increase the unknown word curse. Furthermore, a considerable context can not be considered, like location, friendship, age, and user genre, making it a significant challenge to mix contextual and tweet information.

Xiaohua Liu et al. proposed a KNN classifier with a CRF model for extracting Named Entity from Tweets that are retrained every 600 instances. The KNN considers global information in Tweets by analyzing each word as a window of 5. The middle one is the word in consideration, and the other 4 are the two previous and two followings. The CRF takes advantage of the fine-grained information in each word using three types of features: orthographic, lexical, and gazetteers related. Summing it up, the model obtained an *F*-Measure of 80.2% in a data set of 12,245 English Tweets. They also showed other results concerning the use of KNN. It adds a 5% performance increase in the model. When comparing it with other models, the trade-off in efficiency retraining is not enough for the performance buff [24].

Chanchal Suman et al. decided to take advantage of the images and URLs in tweets and use them as features. The model was composed of word, character, hand-crafted,

URL, and image features. The URL features were considered since users tend to use URLs as a source, so analyzing can help disambiguate the entities. Similarly, like URL, images can have the same impact. Images may contain person, location, graphics, brands, etc.. that give contextual information for the model to consider. In the end, the model proposed was composed of a CNN for character features; BI-LSTM for Word features; the hand-crafted rules; a VGG model was used to extract the images features (VGG is CNN model for image recognition); and for the URL they summarized the information from the web page and then pulled some entities. In the end, they reported an F-1 of 72%. Unfortunately, the impact of the URL and image features was not very beneficial, but it comes from the fact of low number of tweets having images or URL [44].

BERT is a DL model based on transformers (2.2.4) developed by Google and used in the most famous web browser. Its best feature is the bidirectional model being applied to all layers. It was pre-trained with two unsupervised tasks: a Masked LM that, as the name suggests, mask part of the input tokens and tries to predict the masked input; a Next Sentence Prediction is a task that predicts the next sentence. This method was chosen since the relationship between tasks is beneficial in different NLP tasks as Question Answering. From a NER Point of view, it is easy to understand that the pre-trained model will benefit NER. To solve Next Sentence Prediction, the model needs to identify the right entities presented in both sentences. After this step, it is possible to combine the Bert output with other extracted features as an input to a final layer or another kind of model as CRF or even to Fine Tune BERT for NER tasks. Both possibilities were taken in consideration and BERT Presented a F-Measure of 92.8% [10].

Bert was the kickstart for a new revolution in neural models. In the following years, Bert inspired multiple researchers to create new versions of it.

DistilBERT is a distilled version of Bert with 97% of Bert's performance with only half of the parameters required. For this achievement, it introduces a triple loss combining language modeling, distillation and cosine-distance losses while being 60% faster [40].

Robustly optimized BERT approach (RoBERTa) is a retraining of BERT with improved training methodology, 1000% more data and compute power. It removed the Next Sentence Prediction task from BERT's pre-training, and instead of the ordinary Mask LM, it has dynamic masking where the masked tokens change during the training epochs. RoBERTa uses five datasets in its pre-training, making a total of 160 GB [25].

The XLM is a Cross-lingual Language Model pre-trained in three tasks: a next token prediction, a masked language modeling, and a Translation Language Modeling extended to multiple languages. The two main differences from Bert are that each training sample consists of the same text in two different languages, and it has the language ID and the positional embedding of each token in its metadata, which helps in learning the relationship between different languages [20].

With the developments of XLM and RoBERTa, a combination of both was made, creating XLM-RoBERTa. It has 2.5 terabytes of texts in his pre-training with Masking LM task. Once it did not contain parallel text data, the Translation Language Modeling was

dropped from the training data. This approach performed better than XLM and multi-language Bert, especially in low resources languages [8].

A Lite BERT (ALBERT) is another version of Bert with three differences. First, it has the same number of parameters for all layers, reducing the size of the model. Second, It reduces the embeddings size instead of keeping it the same as the vectors that are passed between the encoder layers. Finally, ALBERT uses Sentence Order Prediction, which means whether two sentences appear consecutively or not, instead of Next Sentence Prediction [21].

The DeBERTa model has two architectural changes. First, each token is represented with two vectors instead of one. The first vector represents the content while the second one the relative position. With this change, the self-attention layers can better model the dependency of nearby token pairs. On the other hand, the absolute position of a word is also important. For this reason, an absolute position embedding is added just before the softmax layer of the token decoding head [15].

The authors of ELECTRA understood that MLM had a limitation that at each step only the masked token are updated. Then, they developed a two-model approach. The first model works as a standard MLM and Predicts Masked tokens. The second model takes as input the output of the first model and predicts which of the tokens were originally masked. This change makes the training 30 times more efficient. To use ELECTRA in other NLP tasks the second model is fine-tuned while the first remains the same [7].

Jie Yang et al. made an article testing if it is worth it to have representations of a word and/or character sequences. Each sequence was tested in the architecture of CNN or LSTM. Their work concludes that character-level features can improve the model's performance, especially on disambiguation words. Yet, in character level, the CNN model, even with a similar arrangement, has an advantage over LSTM. Meanwhile, in Word Level, the LSTM had better performance overall. Finally, they concluded with testing the effectiveness of a CRF model to decode the LSTM layer, and as in other authors' reports, it gives a better performance in NER [46].

J. Giorgi et al. researched Biomedical Named Entity Recognition (BNER). The difference between BNER and NER is the type of entities that each task aims to extract. BNER aims for genes, diseases, and species. The model developed uses character and word embeddings. The neural network is composed of BI-LSTM nodes that feed a prediction label, responsible for calculating the probability of each class to each word. Finally, a CRF model outputs the most likely sequence of labels. The model was the first pre-trained with a Silver standard corpora, which are documents that were only classified using multiple BNER systems instead of human resources, so they have lower quality but a higher number of labeled entities. Then it is tuned with Gold standard corpora that are handed labeled data. They reported that transferring learning is better in small sets making an 11% reduction of errors. [12].

Yanyao Shen et al. made a Deep Active Learning model (2.2.4) with a CNN Word encoder and character encoder combined with an LSTM tag decoder. They experimented

with three different types of active learning:

1. Least Confidence: sort examples by ascending order to the probability assigned by the model
2. Maximum Normalized Log-Probability: This method normalizes the Least Confidence. This is important since it had the tendency to select longer sentences
3. Bayesian Active Learning by Disagreement: samples according to the measure of uncertainty

When comparing the performance of the three different techniques with a normal baseline, the models achieved a 99% performance of the best NN models using only 24.9% of the training data in English. They also studied the impact of different data genres in the data set. So they created three data sets, one containing all genres, one mostly made of the newswire genre, and the last one is the opposite of the newswire one. The model with all kinds of data had the best performance, confirming that a diverse data set is mandatory for higher performance. Still, the most exciting conclusion is that the learning model, the Maximum Normalized Log-Probability, in the newswire case select a lower number of newswire sentences than the other two models. While the no newswire model was the one that picked more newswire sentences of the three without any knowledge of the type of data. In conclusion, this model did not just choose the more uncommon genre but also understood the different genres in the data set. [43].

Alan Akbik et al. composed a model to mitigate the problem of unknown words. Typically, entities are not anonymous to a reader with the context in mind, so an unfamiliar word to appear was already explained in the document or is well known in the domain. To mitigate this problem, the simple solution proposed is to pool the word embedding with a memory word embedding. In other words, the embedding will be made of a mix between old embeddings and a new one calculated at the current time stamp. In conclusion, this model presented F-1 of 93.18% in CONLL-03 EN data set and proved to be relevant in other languages such as German and Dutch with 88.27% and 90.44% in the CONLL-03 DE CONLL-03 NL, respectively [1].

Zhilin Yang et al. presented a new model for sequence tagging based on gated recurrent units and conditional random fields. There were two versions of this model. The first was a multi-task model for NER, POs, and Chunk in English, with a shared GRU component that takes as input a character GRU model and word embeddings. Then each task had its own CRF layer that would produce the desired output. The second was a multi-lingual NER model for English, Spanish and Dutch. The model had a shared character embedding that would feed each language's specific GRU and CRF model combined with an independent language word embedding model. This research obtained state-of-art results in a different task, NER, and Chunk, and in other languages reporting F-1 measures of 91,2% in NER English, 85.19% in NER Dutch, and 85.77% in NER Spanish [47].

Besides this model previously presented, big tech companies have various private approaches such as Amazon Comprehend or Comprehend Medical for medical proposes, Watson Natural Language Understanding from IBM, Google Cloud Natural Language API, etc. Unfortunately, there is no information on how these models work and how they were trained. Thankfully there are some open license NLP tools, like NLTK, Spacy, or DBPedia Spotlight. While some have an open API and straightforward implementation, others still hide their secrets. No matter what, these tools can help to develop a new model.

3

Implemented solution

This chapter discuss the three models that were developed. The three approaches distinguish themselves in their design, but keep a common characteristic: A word is always classified taking into account the sentence context it is in. Also, a word may provide more then one classification in the same sentence.

The first one, which I called *Pipeline* for simplicity, is a classic approach based on a set of features designed to solve a classification problem, where objects has to be characterized according to their features. Using features of different kinds, some are obtained with the aid of the class knowledge, others purely based on statistics, or even obtained from NN. However, all are language-independent and do not consider the dataset's classes. Then a traditional SL algorithm classifies each word.

The second and third methodologies are transformer-based. While the second one consists of full implementation of a neural network that is only trained with the dataset, the third one finetunes the dataset in a pretrained BERT version. Both are encoder-only types, transforming the word through a tokenizer to be classified by the Transformers neural network layers. The full implementation of all models can be found in the following link https://drive.google.com/drive/folders/1_QiBI4f7LEjYx2NXD2vmhNnaIoDe-qV0?usp=sharing

3.1 Pipeline

The Pipeline is made of statistics and neural network metrics has referred above. The process starts by applying the Extraction of Multi-grams algorithm to all the datasets. This algorithm provides three crucial features:

1. The N-grams can be extracted from the dataset, which is the primary purpose of the algorithm.

2. The neighbor's counter for each word is the number of different neighbors each word has, being a neighbor, all the words before and after it.

3. The frequency of each word in all the dataset and in each document.

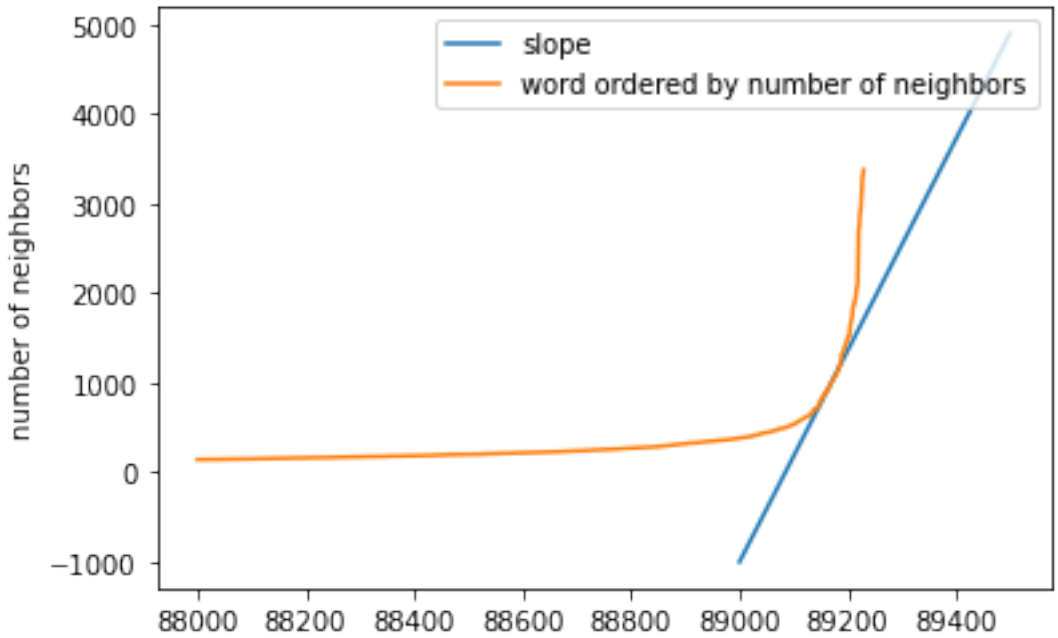

Figure 3.1: Graph representing how the slope is done to find the stop words

Then the Pipeline creates the stop word list by sorting the words by the neighbor's counter, creating an elbow function. This is possible since most words have a similar and low number of neighbors, except the stop words, which has a clearly higher number of neighbors suggesting a natural frontier. Looking through the function and analyzing the tangent periodically will be almost constant until reaching the elbow, where the tangent increases very fast. This sweat spot, with the tangent higher than 11, distinguishes the common words from the stop words. This can be visualized in Figure 3.1 where all the words in this vocabulary are ordered by their number of neighbors.

At this point, the Pipeline can create a few more features that are, say, purely statistical such as:

- If all the letters in the word are upper case.
- If the first letter of the word is in upper case.
- If there is no version of that word with all letters in lower case.
- The length of the word.

While these features are straightforward, they help to distinguish the NE from the Outsiders. Most of them, independent of the dataset, start with an upper letter or are longer. It is vital to check if there is a word version that does not begin with a capital letter, as it may indicate that the occurrence was just the start of a sentence.

The third step is to compute a *bag of words* for each word. It uses a window of size nine, counting the four words before and after the word. This computation is pivotal once it is used in two features. The first feature is the number of words that appear in

all windows of a given word. Looking at the following sentence *Socrates was a Greek philosopher and professor of Plato and Xenophon*, Table 3.1 represents how this process is done to each word.

Table 3.1: Bag of Words .

Word	Counter	Bag of Words
Socrates	4	was a Greek philosopher
was	5	Socrates a Greek philosopher and
a	6	Socrates was Greek philosopher and professor
Greek	7	Socrates was a philosopher and professor of
philosopher	8	Socrates was a Greek and professor of Plato
and	9	was a Greek philosopher professor of Plato and Xenophon
professor	8	a Greek philosopher and of Plato and Xenophon
of	7	Greek philosopher and professor Plato and Xenophon
Plato	6	philosopher and professor of and Xenophon
Xenophon	4	professor of Plato and

The second feature from the *Bag of Words* is made to distinguish the different NE in the dataset. Thus, Considering the Person class, the Pipeline starts by creating a list with words that appear near NE classified as Person but do not appear near other types of NE . In this case, the list would be something like this: *son*, *biographer*, *husband*, *and others*. To create this list, the Pipeline takes into consideration the following metrics:

M1 is the number of times the word is in the neighborhood of any word;

M2 is the number of times the word is in the neighborhood of an Outsider;

M3 is the number of times the word is in the neighborhood of a specific type of NE . This metric is calculated for each class of the datasets.

With these metrics, the Pipeline computes the following expression for each word W :

$$PercentageperClass(W) = \frac{M3}{M1 - M2} \tag{3.1}$$

The *M1-M2* represents the number of times a word appears in the neighborhood of any NE . In contrast, the *M3* represents the number of times a word appears in the neighborhood of specific class of NE . Before the calculation of the given formula, the Pipeline selects the top 1000 words with a higher M1. This acts like a filter, preventing more sporadic words from being on the list. The final step is to select all the words with a higher *PercentageperClass(W)* than 60%. The remaing will be cut off the list as they are too generic. In the end, the Pipeline has a list for each class of the dataset. Each list correspond to the most common word's neighbors in a class, as seen in Table 3.2.

Now that the Pipeline has the lists, it will use them to create a feature. For each word, it will compare the word's bag of words to each list. If a word in the list is also in the

Table 3.2: List of words that are the most common word's neighbors per class

Location	Person	Organization
mi	Pierre	Network
rivers	Friedrich	Committee
km	von	Association
borders	poet	Commission
coast	Jean	Corporation
northwest	friend	Conference
northeast	theory	Group
southwest	actress	Office
flows	Chris	Society
district	marriage	Federal

bag of words, then it is a True match. Else it is a False match. Let's consider the word *Marie* and his bag of words is *a*, *the*, *of*, *and*, *married*, *son and Peter*. Matching this bag of words with the list in the Table 3.2, it would turn out as [0,1,0]. The 0 for the unmatch with the Location list, 1 for the match with the Person List and 0 for the unmatch with the Organization list .

In the fourth step, the Pipeline calculates the stems of each entity. Since it does not use any language base, all the stems are calculated throw statistics. For this achievement, it breaks each word making multiple combinations, from its full length to multiple subsets with at least five letters, as shown in Table 3.3.

Table 3.3: Creation of stems

Length	8	7	6	5
America		America	Americ ; merica	Ameri; meric; erica
Village		Village	Villag ; illage	Villa; illag ; llage
Waterloo	Waterloo	Waterlo;aterloo	Waterl;aterlo; terloo	Water;aterl;terlo; erloo

The Pipeline counts how many times each subset appears for each word. With this information, each entity is assigned to the subset that has a higher counter considering all words. Looking at the word *Waterloo*, for example, it is represented with the stem *Water* as shown by the Table 3.4.

Table 3.4: Occurrences of Waterloo's Subsets

Subset	Waterloo	Waterlo	aterloo	Waterl	aterlo	terloo	Water	aterl	terlo	erloo
Counter	2	2	2	2	2	2	29	10	7	6

Some words that are also represented by the stem *Water* are Watergate, Waterway, Waterlooville, Waterberg, and Waterman.

Then the Pipeline applies a similar idea as the bag of words to create new features.

For each class, the Pipeline creates a list with the most common stems per Class. To accomplish this, it applies the same function as (3.1) but in this case:

M1 is the number of times a stem represent a word;

M2 is the number of times a stem is representing an Outsider;

M3 is the number of times a stem represents a specific class. The Pipeline calculate this metric for each class.

In order to create a more balanced list, it only considers the 100 most used stems of each class and for a stem to be selected, it needs to have more than 60%, according to the formula. In the end if the stem of a word is part of the list then it is a True match else is a False one.

The last features to be created are based on Word2Vec. The Pipeline can create a unique model to any language according to the dataset. Then, it realizes an analogous procedure, like the stems or the bag of words. In this case, it compares the most similar words of an entity to the classified words. By classified words I mean all words that whose class(Named Entity) is known by supervision. Considering the previous case of an English dataset with Location, Person and Organization classes, analyzing the entity *Portugal*, which has *Spain*, *Lisbon*, *France* has most similar words, it would be represented as *[0,3,0,0]*, that is : 0 means that Portugal has no similar classified word as Outsiders; 3 means that there is 3 classified words similar to Portugal in class Location;0 means that Portugal has no similar classified word as Person;0 means that Portugal has no similar classified word as Organization.

To sum it all up, the Pipeline uses six plus three times the number of Classes. So in a dataset where the classes are Location, Person, and Organization, it would have 15 features:

1. If all letters are upper case
2. If it starts with an upper case letter
3. if there is not a version of the word in lower case
4. the word's number of letters
5. the size of its bag of words
6. if the stem is part of the Location stems list
7. if the stem is part of the Person stems list
8. if the stem is part of the Organization stems list
9. if there is a word in its neighborhood that is part of the Location's neighborhood list

10. if there is a word in its neighborhood that is part of a Person's neighborhood list
11. if there is a word in its neighborhood that is part of the Organization's neighborhood list
12. if a similar word is an Outsider
13. if a similar word is a Location
14. if a similar word is a Person
15. if a similar word is an Organization

The last step is to train the training data according to these features. Multiple algorithms were tested being themselves SVM, Naive Bayes, Rigdge Classifier and MLPClassifier. The algorithm chosen was the MLPClassifier as it was the one that presented better results in all the tests made.

3.2 Transformers

The second approach was made through a transformer-based model with an Encoder-only architecture. In Subsection 2.2.4.3, there is a high-level explanation of a Transformers model. In this Section, there will be an extensive explanation of the transformers encoder.

The transformer encoder starts with transforming each word into a token and position embedding, which are then mixed to create a single one. The token embedding maps each word to an integer. In this case, all the words are converted to lower case to reduce the vocabulary dimension and only take the 30 thousand most common words. The others are considered unknown. The idea behind the positional embedding is if there is a pattern correlated to the position of an entity, this layer can learn it. It is similar to the token embedding but uses the position index instead of the token ID as input.

Then comes the essential part of the transformers, the self-attention layers.

The attention comes from considering the whole sentence to produce the embedding. It is possible since it uses weighted vectors from different word relations to compute the final embedding. Furthermore, the self comes from the weights being calculated to all states in the same set.

A self-attention layer is implemented in 4 steps:

1. Project each token embedding in three vectors: *query*, *key*, and *value*.
2. Compute the dot product of the *query* and *key* to create the attention scores.
3. Normalize the variance of the attention scores, which is done to mitigate the ambiguity problem, and normalize it so that each column values sum to 1, creating the attention weights.

4. Multiply the *value* by the attention weights in order to update the token embeddings

Then multiple attention layers are stacked together, each with its query, key, and value projections to create Multi-head attention. It is done to mimic the filters in CNN, where each head focuses on one aspect of similarity between the words. With multiple heads, the model can consider multiple similarities recognizing word relations. In the end, the token embedding is concatenated to form a final embedding. The last part of the block is the Normalization layers that normalize each input in the batch to have zero mean and unity variance.

Finally, two more layers with droupouts are added to the transformer block. An initial feedforward layer with a ReLU activation (2.2.4.5) and a second feedforward layer with a Softmax (2.2.4.5) will give the final classification, as seen in image 3.3.

This model uses crossentropy as the loss function. As this model has a class for the padded tokens, the loss function does not consider it when calculating the loss.

3.3 BERT Fine tuning

The last approach is also based on transformers, but it fine-tunes a pre-trained BERT Model [46] with the training data. The BERT base uncased version would be used if the data were in English. This version is an Encoder-only that was pre-trained on raw text for masked language modeling and for next sentence prediction tasks that help the model to learn an inner representation of the English language. BERT was trained on 4 clouds TPUs in Pod configuration (16 TPU chips total) for one million steps with a batch size of 256. The sequence length was limited to 128 tokens for 90% of the steps and 512 for the remaining 10%. The optimizer used is Adam with a learning rate of 1e-4, β1=0.9, and β2=0.999, a weight decay of 0.01, a learning rate warmup for 10,000 steps, and linear decay of the learning rate after. The data came from the BooksCorpus [48] with 800M words and English Wikipedia with 2,500M words

The BERT multilingual base model is used if the data is in any other language. This version was pre-trained on 104 languages with the largest Wikipedia. Not all the languages would have the same amount of data, as it is easier to find data in English than in Galician. To make a more balanced data set, an exponential smooth was done with a factor of S, $S = 0.7$. It turned high-resource languages to be under-sampled and low-resource languages to be over-sampled. For tokenization, the model uses a 110k shared WordPiece vocabulary [10].

In both models, tokenization is done in the same way in three steps:

Text normalization lowercase the input and removes the accents.

Punctuation Splitting breaks apart the punctuation from the rest of the sentence through white spaces on both sides.

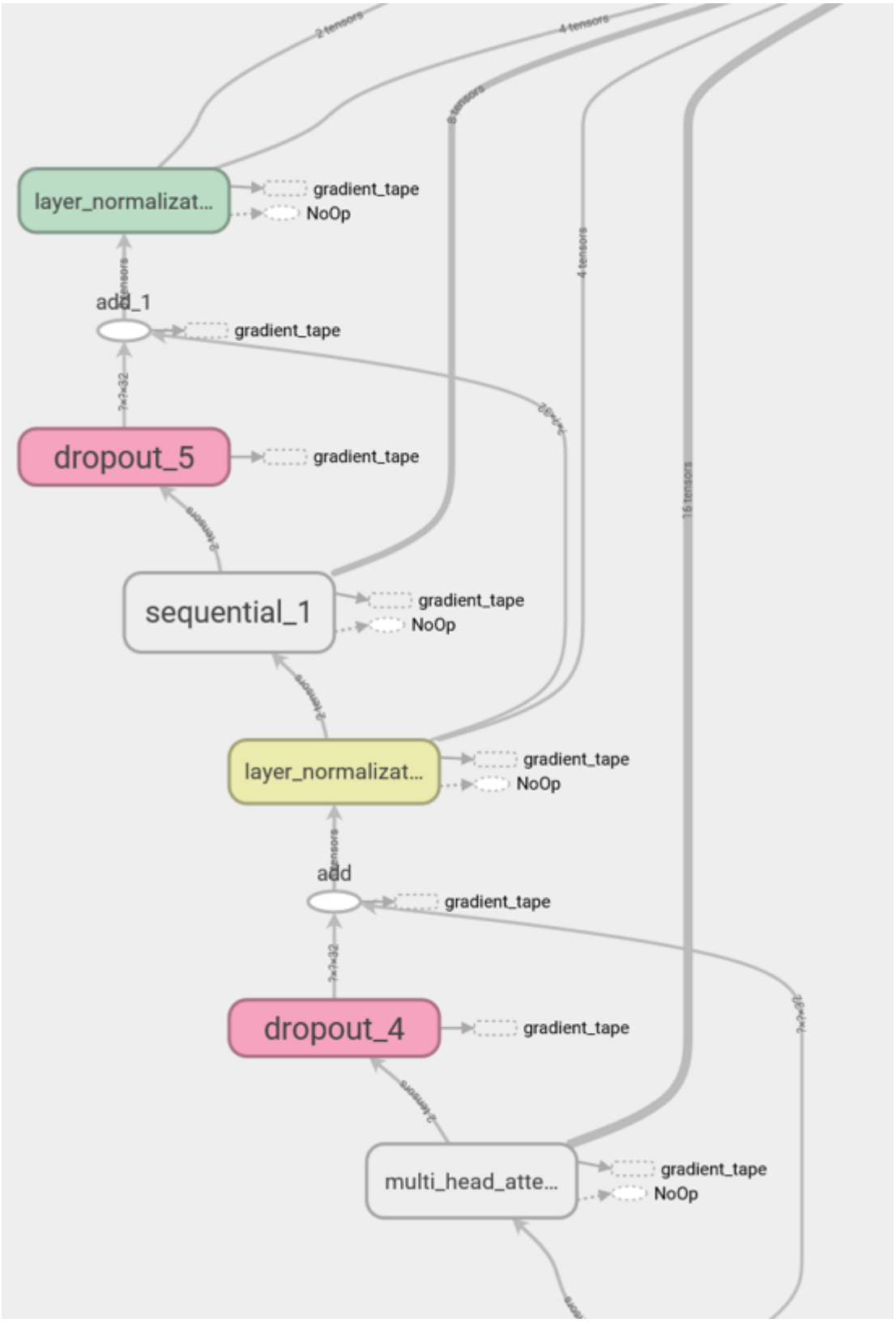

Figure 3.2: Graphical representation of the transformers block

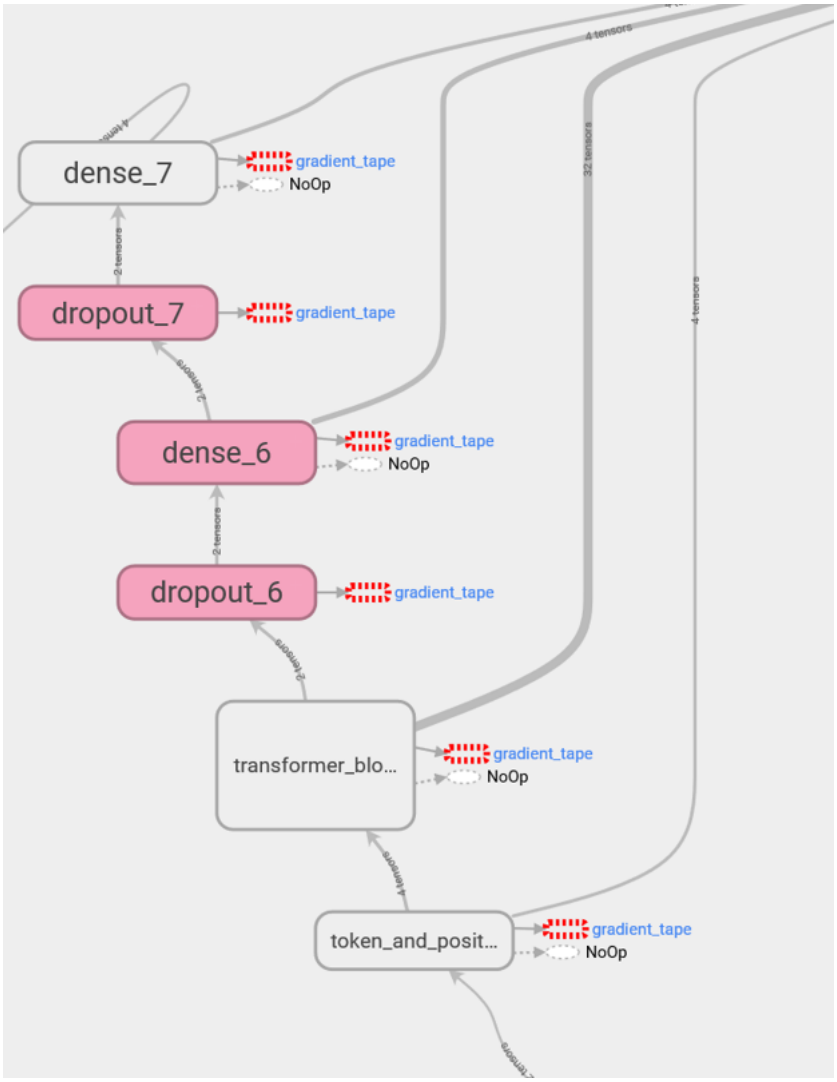

Figure 3.3: Graphical representation of the transformers neural network

WordPiece tokenization apply whitespace tokenization to the input and apply WordPiece tokenization to each token separately.

Since both models are well trained and give essential knowledge on words' relationship, they can be used as a foundation for other models. With finetunnig is possible to define a final layer that classifies the input according to the data. It is also known as learning transferring.

Like any other model, it is biased and will affect any fine-tuned version.

4

Results

This chapter aims to analyze and discuss the results using the different solutions presented in chapter 3. It starts showing the datasets that are going to test the multiple methods. Then a Section presenting the results for each data. This chapter ends with the discussing of the results.

4.1 Datasets

Since this thesis aimed to evaluate how the different technologies react with distinct datasets, seven datasets were chosen according to the three dimensions: size, language and type of entities to be extracted. Some datasets are classified with BIO tags. However, this study discards BIO tags as it was not in the scope of this research. So an entity classified as B-CLASS would be considered as CLASS. Words that do not belong to any type of NE are considered Outsiders("O").

This research studied 12 types of NE , but not all datasets contain all classes. This explains why some classes may be contained in others. The following list presents a definition for all classes.

Person represents a person's name.

Local represents the name of a city, town, river, ocean, mountain and every other type of geographical reference.

Organization represents an organization, public or private.

MISC represents Miscellaneous events, products, and nationalities.

Value represents any kind of numerical value.

Date Represent dates, as days, months ,or years.

Title represents the title of an oeuvre, such as a book or a movie.

Thing represents objects.

Event represents events as conferences or festivals.

Abstraction represents something that can not be physically represented.

Other represents any other NE that is not represented by Person, Local, Organization, Value, Date, Title, Thing, Event, or Abstraction.

Species represents a specie of a living organism.

Medical represents a disease.

This research studies seven datasets, each with different attributes from the other, creating an opportunity for deep research. The following list presents the datasets:

English dataset from wikipedia with four types of NE as: Location, Person, Organization and MISC [36].

Spanish dataset from WikiNEuRal exploits the texts of Wikipedia and introduces a new methodology based on the effective combination of knowledge-based approaches and neural models, together with a novel domain adaptation technique, to produce high-quality training corpora for NER [45]. This dataset has four types of NE : Location, Person, Organization and MISC .

Portuguese dataset with nine classes: Outsiders, Person, Organization, Location, Value, Date, Title, Thing, Event, and Abstraction from multiple genres with origin in every dialect of Portuguese. [41].

Swedish dataset was bootstrapped from Swedish gazetteers and manually corrected/reviewed by two independent native-speaking Swedish annotators. It contains four types of NE as Person, Location, Organization, and MISC.

Twitter dataset written in English made of tweets that was annotated by a combination of NLP experts and crowd workers. The tweets are sampled across different regions, temporal periods, and types of Twitter users. It contains three types of NE : Person, Location, and Organization [9].

Species dataset written in English with only one type of NE that is Species.

Medical dataset written in Spanish from oncological clinical case reports, manually annotated and mapped by clinical experts to a controlled terminology. The dataset only has one type of NE , Medical [29].

Table 4.1 has a statistical representation of each dataset, reporting the total number of tokens, the size of vocabulary, the number of phrases, the number of classes, the median number of tokens per phrase, and the median number of NE per phrase. Table 4.2 describes the number of NE per class in each dataset. Table 4.3 presents the number

Table 4.1: Dataset Status

Dataset	Tokens	Distinct tokens	Phrases	Classes	Tokens/phrase	NE/phrase
English	2.2M	92335	87138	5	24.74	3.47
Spanish	2.4M	99597	95478	5	24.46	3.12
Portuguese	160K	21747	7524	10	21.32	2.42
Swedish	155K	26140	9036	5	17.19	0.90
Twitter	120K	26951	7339	4	16.47	2.01
Species	211K	17580	8196	2	25.84	1.04
Medical	1M	36511	48730	2	22.14	0.75

Table 4.2: Named Entities per Class

Dataset	Person	Organization	Local	MISC	Species	Medical	Value
English	88476	54789	79846	79003	-	-	-
Spanish	90408	42835	95606	68890	-	-	-
Portuguese	3881	3502	3383	-	-	-	1876
Swedish	3976	1519	1797	820	-	-	-
Twitter	6958	4448	3357	-	-	-	-
Species	-	-	-	-	8546	-	-
Medical	-	-	-	-	-	36731	-

Dataset	Date	Title	Thing	Event	Abstraction	Other
Portuguese	1615	1316	463	625	1403	140

Table 4.3: Unique Named Entities per Class

Dataset	Person	Organization	Local	MISC	Species	Medical	Value
English	16464	8783	12199	14370	-	-	-
Spanish	19231	5567	8119	11760	-	-	-
Portuguese	1539	1191	1107	-	-	-	344
Swedish	2077	802	596	644	-	-	-
Twitter	3465	2081	1460	-	-	-	-
Species	-	-	-	-	1840	-	-
Medical	-	-	-	-	-	2517	-

Dataset	Date	Title	Thing	Event	Abstraction	Other
Portuguese	362	545	549	314	689	140

of unique entities per class In both tables, the data is the train, test, and validation split merge.

When comparing the different datasets, it is perceptible that the differences and similarities create good research opportunities. English and Spanish datasets are similar

in size but not in the same language. English and Twitter datasets are in the same language but distinguish themselves in the size of the corpus and the sentences. The Species and Medical datasets are equal in the number of classes but have different sizes. The Portuguese dataset can also study the effect of the number of types.

4.2 Results

This Section presents the metrics for each method in the different datasets. The English, Spanish, Portuguese and Swedish datasets were tested in two forms, the first considering only the NE of Person, Organization and Location (POL) and the second with all the classes belonging to it. To accomplish this, all NE that had another classification were considered Outsiders.

In this research, I used the Exact-match evaluation (2.3.2), where for an entity to be considered well classified, it only requires the correct class. It is a direct result of removing the BIO tags from the classes and discarding the order and where an N-grams starts and finishes.

Before discussing each dataset individually, it is essential to mention the different results between the Macro and Micro metrics. This happens in all the datasets because there is a massive discrepancy between the Outsiders and the other classes in terms of representation. Looking at the English dataset, there are a total of 302K NE and 1.9M Outsiders entities. This discrepancy means that the Micro results will be similar to the Outsiders result, while the Macro results have a more balanced representation of each type. It does not mean that one metric is more relevant than the other. Both are taken into consideration when analyzing the results.

4.2.1 English

The English dataset is one of the most extensive datasets studied in this research. It is made from Wikipedia and has a formal style and a comprehensive set of topics. It contains a higher number of NE per phrase, with Location having the most representation. NE only represent 13% of the tokens.

4.2.1.1 English POL

For this test, the MISC NE were removed, so that in the future comparisons can be made between these dataset and others. MISC entities represent 79k of NE . Then the number of NE per phrase is much smaller, being, in fact, 2.56.

The Pipeline had 15 features. The Transformer model was trained during 10 epochs, finishing with a loss of 0.046. The BERT model was trained during 3 epochs and presented a loss of 0.038.

Table 4.4: Precision Results in English dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9794	0.7906	0.575	0.5	0.7112	0.9508
Transformer	0.9769	0.7567	0.6315	0.6376	0.7507	0.9657
Bert	0.9647	0.8021	0.7071	0.7991	0.8182	0.9542

Table 4.5: Recall Results in English dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9746	0.8292	0.5348	0.8	0.7847	0.9508
Transformer	0.9910	0.3414	0.5714	0.6376	0.6354	0.9657
Bert	0.9961	0.4604	0.5265	0.5815	0.6411	0.9542

Table 4.6: F-Score Results in English dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9770	0.8095	0.5542	0.6153	0.7390	0.9508
Transformer	0.9839	0.4705	0.6000	0.6376	0.6730	0.9657
Bert	0.9801	0.5850	0.6036	0.6732	0.7105	0.9542

The results are the expected in the English POL dataset, with the Transformer having the higher F-Score micro. The biggest surprise is the macro results, with the Pipeline having the highest performance. Considering the dataset, which has a high number of tokens, it was expected to perform well in identifying each class. Another conclusion from this test is that both Transformer and BERT present higher Precision than Recall values. Finally, in both Transformer and BERT models, the number of entities does not reflect in the model's performance once the class with the highest number of samples is the one that underperforms (Organization). On the other side, the Pipeline has a more direct connection between the number of samples and their performance, with the Person's class having the higher performance, followed by the Location and Organization.

4.2.1.2 English with all entities

In this case, all entities were tested, presenting a higher number of entities and one more class for the models to classify. The Pipeline had a total of 18 features. The Transformer model was trained during 10 epochs, ending with a loss of 0.085. The BERT model was qualified during 3 epochs and presented a loss of 0.0425

Table 4.7: Precision Results in English dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9890	0.6842	0.6875	0.5882	0.5568	0.7011	0.9243
Transformer	0.9575	0.6	0.5333	0.6666	0.7107	0.6936	0.9321
Bert	0.9493	0.6622	0.6486	0.7691	0.6698	0.7398	0.9286

Table 4.8: Recall Results in English dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9659	0.9069	0.5	0.9090	0.6621	0.7888	0.9243
Transformer	0.9810	0.4390	0.7619	0.7536	0.4479	0.6766	0.9321
Bert	0.9950	0.4787	0.3809	0.5274	0.4532	0.5670	0.9286

Table 4.9: F-Score Results in English dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9773	0.78	0.5789	0.7142	0.6049	0.7311	0.9243
Transformer	0.9691	0.5070	0.6274	0.7074	0.5495	0.6721	0.9321
Bert	0.9716	0.4836	0.5508	0.6257	0.5406	0.6345	0.9286

Once again, the three models present us with similar results. The Transformer and the BERT performed better, with a 93% F-Score. Both these models had better Precision than Recall performance. A good comparison may be between the Local and MISC classes. Both had a similar number of entities, around 79k, but Location has better results than MISC in all models. Location is less vague than MISC, which facilitates the models' prediction task. The Pipeline is again the model with the higher Macro F-Score, with 73%, overtaking both transformer-based models. The last surprising factor may be the low performance of transformer-based models in the Person class, given that it is the class with the highest number of samples. It should make learning this class easier for the models, something that did not come along.

4.2.2 Spanish

The Spanish dataset is the most extensive dataset studied in this research, with nearly one million unique tokens. It is made from Wikipedia and has a formal style, where topics range from geographical knowledge to socialite news. It has double the amount of Location or Person NE than Organization ones. In this case, NE represent 17,4% of the corpus.

4.2.2.1 Spanish POL

In this test, the MISC NE were removed, representing almost a quarter of all NE in this dataset. The number of NE per phrase is 2.39, and Organization's NE are only 18% of NE

, with Person and Location representing 40% each.

The Pipeline had 15 features. The Transformer model was trained during 10 epochs, finishing with a loss of 0.024. The BERT model was trained during 3 epochs and presented a loss of 0.013.

Table 4.10: Precision Results in Spanish dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9478	0.8534	0.6933	0.8284	0.8307	0.9200
Transformer	0.9849	0.9020	0.8547	0.8918	0.9084	0.9757
Bert	0.9732	0.9381	0.9210	0.9474	0.9449	0.9705

Table 4.11: Recall Results in Spanish dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9782	0.8996	0.4731	0.6007	0.7379	0.9200
Transformer	0.9915	0.8773	0.7795	0.8248	0.8683	0.9757
Bert	0.9988	0.5942	0.7568	0.8091	0.7897	0.9705

Table 4.12: F-Score Results in Spanish dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9627	0.8759	0.5624	0.6964	0.7744	0.9200
Transformer	0.9882	0.8895	0.8154	0.8570	0.8875	0.9757
Bert	0.9858	0.7275	0.8309	0.8570	0.8728	0.9705

Table 4.12 shows outperforming results, in the Spanish dataset with three classes, from both transformer-based models, with a 97% Micro F-Score. Both models were once more Precision-wise than Recall but could perform well in both metrics. The Pipeline shows a good performance, achieving some good results that are not enough to compete with the other models. It underperformed in the Organization class, where it could not distinguish this class from the others. One of the reasons for this may be the underrepresentation of this class compared to the Location and Person.

4.2.2.2 Spanish with all entities

In this trial, all entities were considered, presenting an increasing number of entities and one more class for the models to classify. It reduces the Person's and the Location's percentage from 40% to 30%, and Organization's to 14%.

The Pipeline had a total of 18 features. The Transformer model was trained during 10 epochs, ending with a loss of 0.025. The BERT model was qualified during 3 epochs and presented a loss of 0.019.

Table 4.13: Precision Results in Spanisn dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9675	0.8697	0.6909	0.7666	0.7486	0.8087	0.9162
Transformer	0.9853	0.8678	0.8492	0.9127	0.8133	0.8857	0.9688
Bert	0.9661	0.8666	0.8719	0.9334	0.8925	0.9061	0.9586

Table 4.14: Recall Results in Spanisn dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9852	0.9198	0.5504	0.7443	0.6553	0.7710	0.9162
Transformer	0.9858	0.9262	0.8246	0.8437	0.8277	0.8816	0.9688
Bert	0.9987	0.5711	0.7405	0.7846	0.8277	0.7219	0.9586

Table 4.15: F-Score Results in Spanisn dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9762	0.8940	0.6127	0.7553	0.6989	0.7874	0.9162
Transformer	0.9856	0.8961	0.8367	0.8768	0.8204	0.8831	0.9688
Bert	0.9822	0.6885	0.8008	0.8526	0.7982	0.8245	0.9586

The results of the Spanish dataset with all the entities are pretty similar to those reported in the previous Subsection. The transformer-based models report excellent results with more than 95% of macro F-Score. The Transformer model is the winner in this case as it also gave out 88% of micro F-Score, with all classes having more than 80% individually. BERT had trouble identifying each word's class, resulting in a low Recall. However, the words it identified were almost all correct, reflecting a higher Precision metric. The Pipeline could not keep up with the transformer-based models, nonetheless achieved a 92% micro F-Score with a 79% macro. It's a significant problem that the Organization and MISC entities reported less than 70%. On the one hand, this may be due to the fewer samples. On the other hand, the class with the highest representation was not the one with best results and did not achieve outstanding ones.

4.2.3 Portuguese

The Portuguese dataset has the most classes, but on the other side, it does not have the size of the English or Spanish datasets. Another challenge is the low number of samples from some NE , with lower than 1k NE . This dataset has a vast genre distribution, from e-mails to technical papers or newspapers, and different origins as Portugal, Brasil or African countries [41].

4.2.3.1 Portuguese POL

Only Person, Organization, and Location's NE are studied in this test. Although they are only three of the ten kinds, they represent almost 60% of all NE , meaning that there are 1.4 tokens per phrase.

The Pipeline had a total of 15 features. The Transformer model was trained during 10 epochs, ending with a loss of 0.072. The BERT model was qualified during 3 epochs and presented a loss of 0.207.

Table 4.16: Precision Results in Portuguese dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9276	0.7426	0.6910	0.6363	0.7494	0.9024
Transformer	0.9528	0.6672	0.4315	0.5886	0.6600	0.9431
Bert	0.96010	0.5034	0.5423	0.5207	0.6316	0.9456

Table 4.17: Recall Results in Portuguese dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9811	0.4023	0.4049	0.5811	0.5924	0.9024
Transformer	0.9921	0.2402	0.085	0.3927	0.4276	0.9431
Bert	0.9954	0.2708	0.2709	0.2195	0.4392	0.9456

Table 4.18: F-Score Results in Portuguese dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9536	0.5219	0.5106	0.6074	0.6484	0.9024
Transformer	0.9721	0.3533	0.1429	0.4711	0.4848	0.9431
Bert	0.9774	0.3522	0.3613	0.3088	0.4999	0.9456

The Portuguese dataset with three classes shows a discrepancy between the Outsiders and the Person, Organization, and Location classes. The gap between the Micro results and the Macro ones is the result of this discrepancy. While all models reported more than 90% in micro Precision, Recall, and F-Score, only the transformer-based models could not achieve 50% of Micro F-Score. The Transformer could not classify the Organization NE , having a difference of more than 20 percentage points from the other models. The number of samples can not be taken as a reason for this performance once all classes have an equal representation. The Pipeline and the BERT presented more homogeneous results, with the Pipeline showing up with a 65% Macro F-Score. While any model presented outstanding results, it is notorious that the smaller dataset penalized the transformer-based models.

4.2.3.2 Portuguese with all entities

This case studies how the different methodologies adapt to an unusually higher number of classes. Another adversity is the low number of samples from Thing, Event, Abstraction and Other and how the models will perform in these circumstances.

The Pipeline had a total of 36 features. The Transformer model was trained during 10 epochs, ending with a loss of 0.039. The BERT model was qualified during 5 epochs and presented a loss of 0.342

Table 4.19: Precision Results in Portuguese dataset

Model	Outsiders	Person	Organization	Local	Value	Date	Title
Pipeline	0.9285	0.6571	0.6510	0.6901	0.9375	0.8412	0.6571
Transformer	0.9343	0.2877	0.3783	0.4247	0.5791	0.6855	0.1639
Bert	0.9338	0.3482	0.3140	0.4090	0.6057	0.2061	0.4191

Model	Thing	Event	Abstraction	Other	Macro	Micro
Pipeline	0.7142	0.5365	0.6703	1,0000	0.7530	0.8829
Transformer	0.2285	0.2352	0.1012	0.0000	0.3653	0.8886
Bert	0.5	0.2933	0.1	0.0000	0.3754	0.8982

Table 4.20: Recall Results in Portuguese dataset

Model	Outsiders	Person	Organization	Local	Value	Date	Title
Pipeline	0.9679	0.5679	0.5435	0.6666	0.75	0.7361	0.3239
Transformer	0.9615	0.3666	0.1153	0.4321	0.4893	0.4744	0.0212
Bert	0.9936	0.2568	0.2858	0.2443	0.0870	0.0284	0.0959

Model	Thing	Event	Abstraction	Other	Macro	Micro
Pipeline	0.4347	0.5945	0.5809	0.25	0.5833	0.8829
Transformer	0.0295	0.0343	0.0909	0.0000	0.2741	0.888
Bert	0.0036	0.0944	0.0362	0.0000	0.1933	0.8982

Table 4.21: F-Score Results in Portuguese dataset

Model	Outsiders	Person	Organization	Local	Value	Date	Title
Pipeline	0.9478	0.6092	0.5924	0.6782	0.8333	0.7851	0.4339
Transformer	0.9477	0.3224	0.1767	0.4284	0.5304	0.5608	0.0375
Bert	0.9628	0.2956	0.2992	0.3059	0.1521	0.05	0.1561

Model	Thing	Event	Abstraction	Other	Macro	Micro
Pipeline	0.5405	0.5641	0.6224	0,4000	0.6370	0.8829
Transformer	0.0522	0.0599	0.0957	0,0000	0.2920	0.8886
Bert	0.0072	0.1428	0.0532	0,0000	0.2204	0.8982

The Portuguese dataset with ten entities is a different challenge from all the others in this thesis and the results show that both transformer-based models show considerable problems in this test. The BERT model had trouble with four different classes: Date, Thing, Abstraction, and Others. It could not get over 10% of F-Score in any of these. The Transformer had the same problem even if it achieved a good performance in the Value class, it failed in the others classes. The main factor may be the number of entities represented by each class, but it could not explain the performance in the Date class. The main problem besides the representation may be a vague definition of each class and how it reflects on the dataset. The Pipeline had an outstanding performance compared to the other models. Even though they all had a close Micro F-Score, its Macro F-Score is 63%, which is more than double of transformer-based models. The Pipeline performed much better in all classes than the other models, unless at the Outsiders class. This is an excellent example of how much the Outsiders impact the Micro metrics.

4.2.4 Swedish

The Swedish dataset has the lowest number of NE per phrase. Since it is made from news articles, it is expected to have a formal writing style but does not contain a vast text genre. Person's is the most represented entity, with more than double the other classes, accounting for 49% of NE . Organization and Location's entities have an identical share, with around 20%, and MISC's entities represent only 10% of the total.

4.2.4.1 Swedish POL

This test did not consider the MISC entities. On the one hand, Compared with the other datasets, this is the one where the impact is not as notorious as it represents a small number of samples. On the other hand, Person is more than 50% of the NE .

The Pipeline had 15 features. The Transformer model was trained during 10 epochs, ending with a loss of 0.005. The BERT model was qualified during 5 epochs and presented a loss of 0.018.

Table 4.22: Precision Results in Swedish dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9767	0.8230	0.6153	0.6991	0.7785	0.9577
Transformer	0.9920	0.5704	0.6962	0.1045	0.5908	0.9135
Bert	0.9759	0.8474	0.5284	0.8185	0.7925	0.9721

Table 4.23: Recall Results in Swedish dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9905	0.8263	0.3636	0.5895	0.6925	0.9577
Transformer	0.9260	0.6726	0.3791	0.8436	0.7053	0.9135
Bert	0.9992	0.4166	0.1641	0.5132	0.5233	0.9721

Table 4.24: F-Score Results in Swedish dataset with 3 types of Named Entity

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9836	0.8247	0.4571	0.6396	0.7262	0.9577
Transformer	0.9579	0.6173	0.4909	0.1859	0.5630	0.9135
Bert	0.9874	0.5586	0.2504	0.6308	0.6068	0.9721

The Swedish dataset with three classes shows another good performance from the Pipeline. It is the best model in this dataset, with a 95% in Micro and a 72% Macro F-Score. Once again, his performance is related to the number of entities per class, where the classes with higher representation have higher results. The transformer-based model did not have exceptional performance, with their Macro results not passing the 60%. The little difference in outcomes between the two models may have come from the knowledge base of BERT. The last important aspect is that any models presented an acceptable performance in the Organization class, which has fewer samples.

4.2.4.2 Swedish with all entities

This trial classifies all NE present in the dataset. The MISC class challenges the models, as a low representation class is never easy to predict.

In this case, the Pipeline had 18 features, the Transformer model was trained during 10 epochs, ending with a loss of 0.007. The BERT model was qualified during 3 epochs and presented a loss of 0.019.

Table 4.25: Precision Results in Swedish dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9709	0.7886	0.7222	0.8395	0.6923	0.8027	0.9526
Transformer	0.9828	0.7099	0.7567	0.2171	0.3653	0.6064	0.9504
Bert	0.9727	0.8307	0.5862	0.8132	0.3440	0.7094	0.9676

Table 4.26: Recall Results in Swedish dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9918	0.8690	0.4239	0.5151	0.1894	0.5978	0.9526
Transformer	0.9713	0.6200	0.3562	0.7762	0.0935	0.5635	0.9504
Bert	0.9989	0.4046	0.1717	0.5185	0.1576	0.4502	0.9676

Table 4.27: F-Score Results in Swedish dataset

Model	Outsiders	Person	Organization	Local	MISC	Macro	Micro
Pipeline	0.9812	0.8269	0.5342	0.6384	0.2975	0.6556	0.9526
Transformer	0.9770	0.6619	0.4844	0.3394	0.1490	0.5223	0.9504
Bert	0.9856	0.5442	0.2656	0.6332	0.2162	0.5290	0.9676

When analyzing the Swedish dataset with all classes, the results are comparable to the ones with three entities only. Then the Pipeline is the one with the best performance. Although it has a lower Macro due to the MISC class, the other metrics are practically the same. The MISC class is a rough challenge for all models due to its small representation, with only 820 NE . Any models overpassed the 30% F-Score, and they all had the same problem. They could not retrieve the proper entities, which is reflected in the Recall metrics. Lastly, it is interesting that the Transformer model presented a decent Organization F-Score but a lower Location one while the BERT has a higher Location F-Score but a minor Organization value.

4.2.5 Twitter

The Twitter dataset has the most informal writing style of all. It may create new challenges for the models as they may not adapt. However, according to the table 4.2, the dataset has the highest number of unique tokens related to the total number of words. Since Twitter is a social media made of small texts between users, which explains why Person is the most represented class, accounting for almost half of the NE .

The Pipeline had 15 features. The Transformer model was trained during 10 epochs, ending with a loss of 0.046. The BERT model was qualified during 3 epochs and presented a loss of 0.081.

Table 4.28: Precision Results in Twitter dataset

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.8793	0.5663	0.5714	0.6210	0.6595	0.8391
Transformer	0.9105	0.6201	0.1438	0.5304	0.5512	0.8339
Bert	0.8981	0.7004	0.4827	0.6450	0.6816	0.8831

Table 4.29: Recall Results in Twitter dataset

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9685	0.4965	0.2235	0.2251	0.4784	0.8391
Transformer	0.9354	0.1901	0.3263	0.2168	0.4171	0.8339
Bert	0.9940	0.1939	0.2297	0.3180	0.4341	0.8831

Table 4.30: F-Score Results in Twitter dataset

Model	Outsiders	Person	Organization	Local	Macro	Micro
Pipeline	0.9217	0.5291	0.3213	0.3305	0.5257	0.8391
Transformer	0.9228	0.2910	0.1997	0.3078	0.4303	0.8339
Bert	0.9436	0.3037	0.3113	0.4266	0.4963	0.8831

In the Twitter dataset, any of the models makes an notable performance. The Transformer model has the poorest performance of the three. It had some trouble identifying

the Organization class, with a 19% F-Score, but it generally presented disappointing Recall value. The BERT is the model that made the best performance, with a Micro F-Score of 88% and a Macro of 49%. Having in account the circumstances, the BERT model was the one that brought more expectation, considering how it could transfer everything that the BERT base model knows about the words' relationship to a more informal scenario. When comparing its performance to the Transformer model, it is deductible that the BERT model has more knowledge than the Transformer. And even until this moment, it was not that notorious. The Pipeline did not show the best performance of all datasets. However, it still brought considerable results, achieving a 52% Macro F-Score. The Recall metrics fall behind with a poor performance in the Organization and Local class, the least represented.

4.2.6 Species

The Species dataset only has one class in a scientific writing style and a singular genre. While this should simplify the prediction, it does not have a huge dataset. However, it has a lower number of unique tokens, implying that each word has a higher frequency when compared with a dataset of the same size.

The Pipeline has 9 features, the smaller number possible. The Transformer model was trained during 10 epochs, ending with a loss of 0.029. The BERT model was qualified during 3 epochs and presented a loss of 0.027.

Table 4.31: Precision Results in Species dataset

Model	Outsiders	Species	Macro	Micro
Pipeline	0.9918	1	0.9959	0.9919
Transformer	0.9839	0.2658	0.6248	0.9071
Bert	0.9752	0.7813	0.8782	0.9705

Table 4.32: Recall Results in Species dataset

Model	Outsiders	Species	Macro	Micro
Pipeline	1	0.625	0.8125	0.9919
Transformer	0.9179	0.6642	0.7911	0.9071
Bert	0.9945	0.4342	0.7144	0.9705

Table 4.33: F-Score Results in Species dataset

Model	Outsiders	Species	Macro	Micro
Pipeline	0.9958	0.7692	0.8825	0.9919
Transformer	0.9498	0.3797	0.6647	0.9071
Bert	0.9847	0.5582	0.7715	0.9705

The Pipeline outperformed the other in the Species dataset. A 100% Precision in the Species class and a 100% Recall in the Outsiders class results in 88% Macro F-Score and 99% Micro. The Transformer model had poor Precision, while the BERT did not show a great Recall. The most considerable adversity may have been the low number of samples, only having 1.04 NE per phrase, or a lower number of unique tokens may have affected their results. Once again, a considerable distinction between the Transformer and the BERT models shows up, with a difference of 11 percentage points.

4.2.7 Medical

The Medical dataset is similar to the Species dataset, as they both have only one class. However, this dataset is more extensive, with one million words, which should favor transformer-based models. On the other hand, it has the lowest NE per phrase of all datasets, which may impact the models' performance. The Pipeline has 9 features, the smaller number possible. The Transformer model was trained during 10 epochs, ending with a loss of 0.024. The BERT model was qualified during 5 epochs and presented a loss of 0.019.

Table 4.34: Precision Results in Medical dataset

Model	Outsiders	Medical	Macro	Micro
Pipeline	0.9700	0.6625	0.8162	0.9565
Transformer	0.9815	0.9162	0.9489	0.9804
Bert	0.9755	0.8260	0.9007	0.9737

Table 4.35: Recall Results in Medical dataset

Model	Outsiders	Medical	Macro	Micro
Pipeline	0.9842	0.5047	0.7445	0.9565
Transformer	0.9984	0.4725	0.7355	0.9804
Bert	0.9978	0.2868	0.6423	0.9737

Table 4.36: F-Score Results in Medical dataset

Model	Outsiders	Medical	Macro	Micro
Pipeline	0.9771	0.5729	0.7750	0.9565
Transformer	0.9899	0.6235	0.8067	0.9804
Bert	0.9865	0.4258	0.7061	0.9737

In the Medical dataset, the Transformer performed better, achieving 80% Macro and 98% Micro. In this case, the models would distinguish between each other according to their performance in the Medical class. The Transformer had an exceptional Precision, 91%, even though its Recall was not that high. BERT had an acceptable Precision, but the Recall was too low, resulting in a 42% F-Score, the weakest of all. In contrast to the others,

the Pipeline had a better Recall but the most insufficient Precision, not going further than a 57% F-Score.

4.3 Discussion

One necessary analysis is to compare the English and Spanish datasets due to the size of the corpus and the high sample number. Both datasets have more than 2 million tokens and more than 3 NE per phrase and sentence of the same length. However, the results are not as similar as expected. The Spanish dataset performs better in the two case studies, achieving Macro results of over 80% F-Score. While with the English dataset, the models do not go over 75%. The most expected result would be that the English dataset had a more significant performance, particularly in the BERT case, which has a broader knowledge base. However, this did not occur. It does not mean that the multi-language BERT is better, as the dataset's quality or language may have impacted the results, but it must have something into account.

Besides the big corpus, it is essential to compare the small ones. The Swedish and Portuguese datasets can be compared in the three classes test. In this comparison, the results are analogous. The models present better results in the Swedish dataset, but in my opinion, the difference is not enough due to the different languages other than other factors. It is relevant that the Pipeline was the model that presented higher Macro in both cases while not being the one with higher Micro, mainly because it did not have the better result in the Outsiders class.

The writing style is one of the aspects of the study by comparing the English dataset with Twitter one. In both cases, the performance was not excellent, but it exists an evolution from Twitter to English in all models. Once this upgrade happens in all models, passing from 50% to 70%, it can not only be explained by the data size but by other factors, such as the writing style and the length of each sentence, which are intrinsically correlated to the first one.

The Medical and the Species dataset tested how the models behave with one class in a larger or smaller dataset. According to the results, there is no clear sign that more data helped the models. The Pipeline lost eleven percentual points, from the Species to the Medical. The Transformer won fourteen points, and the BERT lost another seven. These numbers are not easy to explain, as it is expected that better results will come from transformer-based models with more data. However, changing from the English BERT to the multi-language version may be the explanation.

When considering how the number of features affects the models, in my opinion, it is not fair to take any consideration. In datasets with one to four classes, if the size of the corpus increases, it also increases the quality of the transformer-based models. The test with ten classes presented a low performance. However, I think this is more related to the lack of samples than the number of classes. The performance would also increase if

each class had the same samples. Besides the transformer-based models, the Pipeline is also stable in changing the number of classes, always presenting decent results.

5

Conclusion and Future Works

This is the final chapter. You will find a brief description of everything done during this Thesis, the conclusion, and some future works.

5.1 Recap

This Thesis seeks to test how the transformer-based models perform in different contexts and to compare them with a model that does not use the transformer architecture. First, I developed three models:

Pipeline based on statistics and neural network features that is language independent using multiple techniques such as Bag of Words, Steeming, W2V.

Transformer Network that is only trained with the dataset and can be adapted to any language.

Fine-tuning a BERT model , which was the English or multi-language version depending on the dataset, that has a bidirectional architecture made of transformers.

Then I selected seven different datasets to test these models according to three important factors: language, size and classes. The results were analyzed, and a few comparisons between the datasets were made to understand better the performance and the best case use for each model.

5.2 Conclusion

After an extensive analysis of the different tests, it is possible to obtain a few final results. It will start by discussing the transformer-based models and comparing them to my Pipeline. The transformer-based models are highly dependent on data. It is not a surprise, as multiple authors have already proved it. The most significant difference between the Transformer and the BERT came when they were tested with insufficient data. BERT performed better in every lower data case except the Portuguese dataset with

all the entities. This is primarily because of its bidirectional architecture of BERT and its pretrain knowledge. On the other end of the spectrum, when more samples were available, the Transformer model performed exceptionally well and kept up, sometimes even overpassing, the BERT model. It occurred in the English dataset with all entities, in both Spanish tests and the Medical. Regarding resources needed to compute the models, BERT requires more from the hardware while being more time-consuming. Concerning language independence, I think both models are equal, and it does not affect the final results, as both exceeded well in different languages. Lastly, concerning the number of features, the number of features, BERT was more affected when increasing the number of classes. In all the datasets, when comparing the Macro from four to three classes, BERT had a more considerable difference than the Transformer.

Designing a Pipeline based on statistical and neural network features that is language-independent is not an easy task. However, this Pipeline can compete with the transformer-based model. Starting with data size, the Pipeline fell behind the other models in big data corpus. In the English dataset, it presented similar results to the other models. In the Spanish dataset, it did not show up. On the other hand, it showed much better results in low data cases, independent of having three, four, or ten classes and independent of the language. Even when only one class was in the test, it got better results in lower data. This is more important than it looks at first. The Medical and Species test have in common the fact of being very specific, thus it is not common to be considered in NER context. These datasets are almost particular, and increasing the data with more samples is much more time and cost-consuming. It is more time-consuming as the creator has to search in a smaller group of texts. It is possible to create a dataset of POL by looking in newspapers, books, emails, and Wikipedia. This particular dataset has a smaller baseline. Instead of looking in newspapers, you must look up scientific papers or journals. And it is more cost-consuming to classify these entities as it requires someone from the field with a high education level which takes more money for their time.

Summing up, BERT will always present better results in the most common cases of NER , which is to identify a few classes in a formal writing style. However, when the model aims to extract other entities or does not have an extensive dataset, I think that BERT may not be the best solution. Every case should be analyzed individually, considering that sometimes the most straightforward path is not the better one.

5.3 Future Works

As a proposal for future work, I think further tests should be made on the transformer architecture and Neural Networks. My first notion is to continue this investigation, comparing these models with NER models made by tech companies. This Thesis only considered the NER task leaving behind other NLP tasks and Machine Learning problems. My second recommendation is to test how well the transformers architecture work in the different environments and where are its limits. The last ideia is to develop transformer

Neural Networks to be used in more problems besides NLP or Computer Vision and to study how it adapts to different issues.

Bibliography

[1] A. Akbik, T. Bergmann, and R. Vollgraf. "Pooled Contextualized Embeddings for Named Entity Recognition". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019-06, pp. 724–728. DOI: 10.18653/v1/N19-1078. URL: https://aclanthology.org/N19-1078 (cit. on p. 28).

[2] K. Arulkumaran et al. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38. DOI: 10.1109/MSP.2017.2743240 (cit. on p. 20).

[3] M. Asahara and Y. Matsumoto. "Japanese Named Entity Extraction with Redundant Morphological Analysis". In: NAACL '03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 8–15. DOI: 10.3115/1073445.1073447. URL: https://doi.org/10.3115/1073445.1073447 (cit. on p. 24).

[4] D. M. Bikel et al. "Nymble: a High-Performance Learning Name-finder". In: *Fifth Conference on Applied Natural Language Processing*. Washington, DC, USA: Association for Computational Linguistics, 1997-03, pp. 194–201. DOI: 10.3115/974557.974586. URL: https://aclanthology.org/A97-1029 (cit. on p. 24).

[5] P. Blunsom. "Hidden Markov Models". In: (2004-08) (cit. on p. 12).

[6] J. S. Bridle. "Training Stochastic Model Recognition Algorithms as Networks Can Lead to Maximum Mutual Information Estimation of Parameters". In: *Proceedings of the 2nd International Conference on Neural Information Processing Systems*. NIPS'89. Cambridge, MA, USA: MIT Press, 1989, pp. 211–217 (cit. on p. 20).

[7] K. Clark et al. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. DOI: 10.48550/ARXIV.2003.10555. URL: https://arxiv.org/abs/2003.10555 (cit. on p. 27).

[8] A. Conneau et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2019. DOI: 10.48550/ARXIV.1911.02116. URL: https://arxiv.org/abs/1911.02116 (cit. on p. 27).

[9] L. Derczynski, K. Bontcheva, and I. Roberts. "Broad twitter corpus: A diverse named entity recognition resource". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 1169–1179 (cit. on p. 40).

[10] J. Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805 (cit. on pp. 26, 36).

[11] G. Doddington et al. "The Automatic Content Extraction (ACE) program-tasks, data, and evaluation". In: *Proceedings of LREC* 2 (2004-01) (cit. on p. 23).

[12] J. M. Giorgi and G. D. Bader. "Transfer learning for biomedical named entity recognition with neural networks". In: *Bioinformatics* 34.23 (2018-06), pp. 4087–4094. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty449. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/23/4087/26676581/bty449.pdf. URL: https://doi.org/10.1093/bioinformatics/bty449 (cit. on p. 27).

[13] I. J. Good. "Rational Decisions". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 14.1 (1952), pp. 107–114. ISSN: 00359246. URL: http://www.jstor.org/stable/2984087 (visited on 2022-09-15) (cit. on p. 21).

[14] A. Goyal, V. Gupta, and M. Kumar. "Recent Named Entity Recognition and Classification techniques: A systematic review". In: *Computer Science Review* 29 (2018), pp. 21–43. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2018.06.001. URL: https://www.sciencedirect.com/science/article/pii/S1574013717302782 (cit. on p. 3).

[15] P. He et al. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. 2020. DOI: 10.48550/ARXIV.2006.03654. URL: https://arxiv.org/abs/2006.03654 (cit. on p. 27).

[16] S. Hochreiter et al. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by S. C. Kremer and J. F. Kolen. IEEE Press, 2001 (cit. on p. 17).

[17] H. Ji and R. Grishman. "Data Selection in Semi-supervised Learning for Name Tagging". In: *Proceedings of the Workshop on Information Extraction Beyond The Document*. Sydney, Australia: Association for Computational Linguistics, 2006-07, pp. 48–55. URL: https://aclanthology.org/W06-0206 (cit. on p. 25).

[18] S. G. Joaquim Ferreira da Silva Gaël Dias and J. G. P. Lopes. "Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units". In: *Session Poster*, (1999) (cit. on p. 9).

[19] *K means Clustering – Introduction*. [Online; accessed 20-January-2022]. 2021-09. URL: https://www.geeksforgeeks.org/k-means-clustering-introduction/ (visited on 2021-09-21) (cit. on p. 14).

[20] G. Lample and A. Conneau. *Cross-lingual Language Model Pretraining*. 2019. DOI: 10.48550/ARXIV.1901.07291. URL: https://arxiv.org/abs/1901.07291 (cit. on p. 26).

[21] Z. Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. DOI: 10.48550/ARXIV.1909.11942. URL: https://arxiv.org/abs/1909.11942 (cit. on p. 27).

[22] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015-05), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cit. on p. 16).

[23] T.-Y. Lin et al. *Focal Loss for Dense Object Detection*. 2017. DOI: 10.48550/ARXIV.1708.02002. URL: https://arxiv.org/abs/1708.02002 (cit. on p. 21).

[24] X. Liu et al. "Recognizing Named Entities in Tweets". In: HLT '11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 359–367. ISBN: 9781932432879 (cit. on p. 25).

[25] Y. Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692 (cit. on p. 26).

[26] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[27] A. McCallum and W. Li. "Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 188–191. URL: https://aclanthology.org/W03-0430 (cit. on p. 24).

[28] A. Mikheev. "A Knowledge-free Method for Capitalized Word Disambiguation". In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. College Park, Maryland, USA: Association for Computational Linguistics, 1999-06, pp. 159–166. DOI: 10.3115/1034678.1034710. URL: https://aclanthology.org/P99-1021 (cit. on p. 24).

[29] A. Miranda-Escalada, E. Farré, and M. Krallinger. "Named Entity Recognition, Concept Normalization and Clinical Coding: Overview of the Cantemist Track for Cancer Text Mining in Spanish, Corpus, Guidelines, Methods and Results." In: *IberLEF@ SEPLN* (2020), pp. 303–323 (cit. on p. 40).

[30] *ML | Spectral Clustering*. [Online; accessed 20-January-2022]. 2019-07. URL: https://www.geeksforgeeks.org/ml-spectral-clustering/ (visited on 2019-07-19) (cit. on p. 15).

[31] D. Nadeau and S. Sekine. "A Survey of Named Entity Recognition and Classification". In: *Lingvisticae Investigationes* 30 (2007-08). DOI: 10.1075/li.30.1.03nad (cit. on pp. 6, 11, 14).

[32] D. Nadeau and P. Turney. "Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity". In: vol. 4013. 2006-06. ISBN: 978-3-540-22004-6. DOI: 10.1007/11766247_23 (cit. on p. 25).

[33] V. Nair and G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077 (cit. on p. 20).

[34] S. Narayan. "The generalized sigmoid activation function: Competitive supervised learning". In: *Information Sciences* 99.1 (1997), pp. 69–82. ISSN: 0020-0255. DOI: https://doi.org/10.1016/S0020-0255(96)00200-9. URL: https://www.sciencedirect.com/science/article/pii/S0020025596002009 (cit. on p. 20).

[35] *NLTK Documentation nltk.tokenize package*. https://www.nltk.org/api/nltk.tokenize.html. Accessed: 2022-02-02 (cit. on p. 10).

[36] J. Nothman et al. "Learning multilingual named entity recognition from Wikipedia". In: *Artif. Intell.* 194 (2013), pp. 151–175 (cit. on p. 40).

[37] A. Olgac and B. Karlik. "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks". In: *International Journal of Artificial Intelligence And Expert Systems* 1 (2011-02), pp. 111–122 (cit. on p. 20).

[38] P. Ren et al. *A Survey of Deep Active Learning*. 2021. arXiv: 2009.00236 [cs.LG] (cit. on p. 20).

[39] E. Riloff and R. Jones. "Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping". In: (1999) (cit. on p. 16).

[40] V. Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2019. DOI: 10.48550/ARXIV.1910.01108. URL: https://arxiv.org/abs/1910.01108 (cit. on p. 26).

[41] D. Santos et al. "Harem: An advanced ner evaluation contest for portuguese". In: *quot; In Nicoletta Calzolari; Khalid Choukri; Aldo Gangemi; Bente Maegaard; Joseph Mariani; Jan Odjik; Daniel Tapias (ed) Proceedings of the 5 th International Conference on Language Resources and Evaluation (LREC'2006)(Genoa Italy 22-28 May 2006)*. 2006 (cit. on pp. 40, 46).

[42] K. Santos. *How DBSCAN works and why should we use it?* [Online; accessed 20-January-2022]. 2017-04. URL: https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80 (visited on 2017-04-01) (cit. on p. 14).

[43] Y. Shen et al. "Deep Active Learning for Named Entity Recognition". In: *CoRR* abs/1707.05928 (2017). arXiv: 1707.05928. URL: http://arxiv.org/abs/1707.05928 (cit. on p. 28).

[44] C. Suman et al. "Why pay more? A simple and efficient named entity recognition system for tweets". In: *Expert Systems with Applications* 167 (2021), p. 114101. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114101. URL: https://www.sciencedirect.com/science/article/pii/S0957417420308551 (cit. on p. 26).

[45] S. Tedeschi et al. "WikiNEuRal: Combined Neural and Knowledge-based Silver Data Creation for Multilingual NER". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021-11, pp. 2521–2533. DOI: 10.18653/v1/2021.findings-emnlp.215. URL: https://aclanthology.org/2021.findings-emnlp.215 (cit. on p. 40).

[46] J. Yang, S. Liang, and Y. Zhang. "Design Challenges and Misconceptions in Neural Sequence Labeling". In: *CoRR* abs/1806.04470 (2018). arXiv: 1806.04470. URL: http://arxiv.org/abs/1806.04470 (cit. on pp. 27, 36).

[47] Z. Yang, R. Salakhutdinov, and W. W. Cohen. "Multi-Task Cross-Lingual Sequence Tagging from Scratch". In: *CoRR* abs/1603.06270 (2016). arXiv: 1603.06270. URL: http://arxiv.org/abs/1603.06270 (cit. on p. 28).

[48] Y. Zhu et al. *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books*. 2015. DOI: 10.48550/ARXIV.1506.06724. URL: https://arxiv.org/abs/1506.06724 (cit. on p. 36).